FACULTAT D'INFORMÀTICA DE BARCELONA

Extracció de patrons de traducció d'UML a WebML

Projecte final de carrera

Enginyeria Informàtica

Enric Sangrà Navarro

16 de gener de 2013

Universitat Politècnica de Catalunya

Director
Carles Farré Tost (Departament d'Enginyeria de Serveis i Sistemes d'Informació)
Presidenta
Cristina Gómez Seoane (Departament d'Enginyeria de Serveis i Sistemes d'Informació)
Vocal
Francesc Tiñena Salvañà (Departament de Matemàtica Aplicada II)

Agraïments

A la Mar, la meva parella, que m'ha ajudat en tot moment animant-me quan el camí semblava massa llarg.

A tots els companys que m'han acompanyat aquests anys i amb els que compartit tant les classes com la diversió. En especial a l'Albert Plana, per compartir el patiment d'aquest últim quadrimestre i l'Alex Álvarez, per a discutir dubtes quan ho he necessitat.

A la meva família que m'ha recolzat per a que pogués estudiar allò que sempre he volgut.

Al Carles Farré per oferir-me la possibilitat de dur a terme aquest projecte.

A la comunitat de WebRatio per resoldre els dubtes durant l'estudi de WebML.

A la resta de gent amb la que m'he trobat durant el camí i m'han ajudat en un moment o altre, ja fos ensenyant-me o preparant-me un cafè cada matí.

Índex

1 Introducció

Traduir significa escriure en un llenguatge allò que ha estat escrit en un altre tenint en compte les restriccions del context, les normes gramaticals i d'escriptura.

En aquest treball en concret volem aconseguir extreure una sèrie de passos per a poder fer la traducció d'especificacions d'UML a WebML. A aquest conjunt de passos junt amb el context i les normes associades els anomenarem patrons.

Així doncs, a partir d'una especificació amb UML podríem, de forma més o menys sistemàtica, generar diagrames WebML amb el mateix significat que els diagrames UML d'origen. Això contant que els patrons fossin complets i abastessin tots els casos possibles.

Tot el procés es durà a terme a partir d'un cas d'exemple. Es traduirà una especificació d'un Fòrum de l'UML al WebML usant l'eina WebRatio. Aquesta ens permet modelar en WebML i a més a més ens genera el codi de manera que podem validar el model per a demostrar que compleix els requeriments.

2 Motivació i objectius del projecte

Al provar per a primera vegada WebRatio em va sorprendre. Tot i que va costar una mica entendre el rol de les unitats i com enllaçar-les correctament, en molt poc temps vaig ser capaç de fer alguna funcionalitat molt bàsica i que tingués el comportament desitjat.

Mentre feia aquestes proves amb l'eina vaig començar a trobar certes relacions entre la manera d'especificar l'UML amb l'extensió web i la possibilitat que oferia WebRatio amb les pàgines i les seves altres unitats per a mantenir una estructura similar.

Això em va donar la idea de que a diferents nivells d'abstracció podria ser possible extreure unes maneres de fer per a la traducció d'un llenguatge a l'altre i viceversa. Com d'algun punt s'havia de començar i començar des de WebML ho veia difícil perquè no coneixia encara el seu correcte funcionament, vaig decidir passar de l'UML al WebML en aquest projecte.

Així doncs l'objectiu principal d'aquest projecte no és nomes poder aprofundir en WebML sinó poder trobar una relació entre l'UML i WebML i definir d'aquesta manera un conjunt de patrons per a fer la seva traducció d'una manera més automàtica sense necessitat de conèixer completament el WebML per a poder generar el codi dels webs a partir de WebRatio.

3 Metodologia

Per tal d'arribar a l'objectiu principal que és l'explicat anteriorment, es seguirà una metodologia basada en certs sub-objectius que es marquen a continuació. D'aquesta manera podem garantir uns bons resultats finals.

1. **Estudi de WebML per a valorar les possibilitats del projecte**
   Abans de començar el projecte i decidir quin seria l'objectiu, es va començar per a utilitzar WebRatio (eina que treballa amb WebML) per a fer algunes proves i veure quina era la seva potència i si es podria extreure alguna cosa de treballar amb el llenguatge.
2. **Especificació del fòrum d'aprenentatge cooperatiu en UML**
   Explicació del fòrum d'aprenentatge cooperatiu, funcionalitats, casos d'ús detallats, model conceptual, diagrames de seqüència de les operacions, UX model i aplicació de l'extensió web.
3. **Elaboració de l'aplicació amb l'eina WebRatio que utilitza WebML a partir de l'especificació creada anteriorment**
   En aquest pas es fa una traducció documentada del fòrum d'aprenentatge des d'UML a WebML.
   WebRatio ens serveix per a veure que el diagrama complirà amb el seu propòsit ja que genera el codi web en Java a partir dels diagrames i és possible executar-lo.
4. **Extracció de possibles patrons de traducció**
   Un cop feta la traducció, es compararan dins del WebML fragments de diagrames que eren semblants en UML per a veure si es segueix algun tipus de patró. En aquest pas potser serà necessària la reestructuració d'algun diagrames UML per tal d'aconseguir una millor semblança sempre mantenint el mateix significat.
5. **Validació dels patrons extrets**
   Un cop extrets els patrons s'intentaran posar en ús amb algun altre cas en petit format per a veure que realment serveixen en el context marcat.

4 Planificació inicial

Tal com s'ha vist a la metodologia de l'apartat anterior, dividíem el projecte en cinc parts.

L'**estudi de WebML** es va iniciar prèviament a l'elecció del projecte ja que s'havia de valorar el llenguatge per a saber si hi havia algun projecte possible i decidir, després d'investigar-lo, quin seria l'objectiu final del projecte.

L'**especificació amb UML** hauria de ser la part fàcil del projecte donada l'experiència d'altres assignatures en aquest aspecte i el coneixement dels artefactes a definir. Tot i així, l'elecció de l'aplicació havia de fer-se pensant en les possibilitats que podria donar a l'hora d'extreure els patrons.

L'**especificació amb WebML** es prometia una mica complicada. Tot i haver fet algunes proves i conèixer certs aspectes de les seves característiques no sabia si podria traduir l'especificació completa amb els coneixements previs.

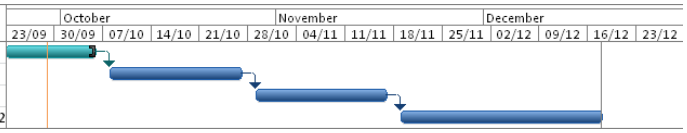
L'**extracció dels patrons** era una de les parts més importants del projecte ja que resumia els objectius d'aquest i encara que semblés fàcil perquè durant la traducció es poden anar trobant i entenent els diferents patrons, definir-los correctament seria un altre tema.

Per acabar amb els patrons, s'haurien de **validar** per a comprovar que són realment útils. Si volem fer-ho a partir d'un altre cas, encara que sigui petit, ocuparà una bona part del temps.

A part d'aquestes cinc parts que marca la metodologia, també s'ha d'afegir una part per a la correcta **documentació** del projecte i dels seus resultats.

La planificació resultant tenint en compte aquests factors és la següent:

ID	Task Name	Duration	Start	Finish
2	Especificació UML	26 days	Sun 02/09/12	Fri 05/10/12
3	Traducció a WebML	15 days	Mon 08/10/12	Fri 26/10/12
4	Extracció de patrons	15 days	Mon 29/10/12	Fri 16/11/12
5	Validació de patrons	21 days	Mon 19/11/12	Mon 17/12/12



Il·lustració 1: Gantt de la planificació inicial del projecte

Al final del projecte es fa una valoració de com ha estat finalment.

5 WebML

WebML és un llenguatge de modelatge per a aplicacions web creat per un grup de recerca del *Dipartimento di Elettronica e Informazione* de la universitat *Politecnico di Milano*.

Aquest llenguatge de modelatge sorgeix per tal de poder canviar la manera de desenvolupar el software web i reduir l'esforç necessari per a crear prototips d'aplicacions. La idea és poder augmentar el nivell d'abstracció per tal de centrar-se més en la creació.

Mitjançant el modelatge, tot el codi generat és usable, és autodocumentat i redueix l'esforç en cost i temps ja que permet generar el prototip en qualsevol moment (WebRatio).

WebML està orientat a webs amb una gran quantitat d'informació, basades en la cerca d'aquesta i el seu tractament. No està centrat en sistemes web lògica molt complexa tot i que també permet desenvolupar-ne. A més a més, dona eines per a poder modificar fàcilment l'estructura, el model de navegació o la presentació de forma senzilla sense afectar a la resta de l'aplicació.

Com a resum, es pot dir que no és un llenguatge fet per a modelar pàgines de poca informació o estàtiques.

Les aplicacions web es poden separar en diverses capes. Una de les representacions que es proposa és la següent:

Presentació

És el nivell que controla com s'organitza la informació d'una pàgina i l'estil amb que es mostra.

L'organització es fa mitjançant una quadrícula o *grid* que permet posicionar les unitats dins la pàgina.

L'estil es defineix a partir de *layouts* que permeten donar un format específic a una unitat concreta per a la seva visualització.

Models d'usuari

En aquest nivell es permet dividir el web en diferents vistes o *site views* de forma que és fàcil crear diferents mòduls per cada tipus d'usuari del sistema i d'aquesta forma delimitar els seus permisos.

Navegació i composició (Hypertext Model)

Aquesta és la capa amb la lògica de l'aplicació. Permet definir les pàgines que tindrà el web, com interactuaran entre elles, quina informació mostraran i quines operacions es permeten. A continuació s'explica cada una d'aquestes parts ja que tindran molta importància durant el projecte.

Les pàgines o *page* indiquen una pàgina de l'aplicació i està composta per unitats de contingut que representen la informació de l'aplicació. D'aquesta forma, a cada pàgina decidim quina informació de la base de dades volem que es mostri a l'usuari. Les pàgines es poden agrupar en àrees.

A les pàgines se'ls poden afegir certes propietats que les transformaran en pàgines amb comportament propi:

- Les *Home Page* són aquelles pàgines principals d'un *site view* i només en pot haver una per a cada vista.
- Les *Default Page* estan dins de les àrees o *Alternative Pages* i són aquelles que es mostraran en el cas d'entrar a l'àrea sense definir cap pàgina com a destí.
- Les *Landmark Page* són unes pàgines accessibles des de qualsevol altre pàgina de la *site view*.

També existeix la possibilitat d'usar pàgines com a contingut d'una altra. Aquestes pàgines s'anomenen *Sub Pages* i permeten crear quelcom semblant als *frames* de HTML. S'utilitza quan és es vol separar el contingut de la pàgina en varis grups diferents.

En el cas concret de voler tenir varies pàgines possibles però només mostrar una, s'utilitzen un altre tipus de pàgina anomenada *Alternative Page*. Una de les pàgines de dins ha de ser una *Default Page* que es mostrarà si no es diu el contrari.

Les unitats de contingut o *content units* són les unitats que defineixen la informació que mostrem a les pàgines. Principalment, una unitat de contingut es representa amb una capsa del tipus d'unitat i l'entitat de la base de dades a la qual representa. A més a més, s'hi pot incloure un selector, un predicat per determinar quins objectes mostrar.

Hi ha catorze tipus d'unitat de contingut diferents. A continuació es nombren les cinc que s'han utilitzat en el projecte i una breu descripció.

- La *Data Unit* és un tipus d'unitat que permet mostrar la informació relativa a una entitat i les seves relacions. Està restringit a una sola instància i incorpora un selector sobre l'atribut clau de la instància.
- La *Index Unit* permet mostrar informació de múltiples instàncies d'una entitat i de les seves relacions. En cas de voler seleccionar només un subconjunt de les instàncies existents és necessari afegir-hi un selector.
- La *Power Index Unit* és igual que l'anterior però en aquest cas es pot definir el tamany dels blocs a mostrar i genera automàticament una paginació per a recórrer totes les instancies de *n* en *n* segons el tamany definit.
- La *Entry Unit* està composta de diferents camps i d'aquesta forma permet crear formularis. A cada camp es pot escollir entre camps de text, de selecció o selecció múltiple i se li pot afegir un tipus de variable.
- La *Multi Message Unit* permet incorporar text dinàmic a les pàgines. Mitjançant *placeholders* es pot deixar informació depenent de valors de variables de l'aplicació.

Les operacions o *operation units* permeten afegir, modificar o esborrar la informació de l'aplicació. Són la base de la interacció amb el sistema i tot i poder estar dins les pàgines no mostren cap informació. Igual que les unitats de contingut, estan relacionades també amb una

entitat o una relació de la base de dades amb la qual interactuen i poden tenir selectors. No necessiten pertànyer a una pàgina per a poder existir.

Totes les operacions compleixen una sèrie de principis:

- Una operació pot tenir més d'un link d'entrada per a proveir els paràmetres d'entrada.
- Una operació segueix un OK Link de sortida en cas de que l'operació acabi en èxit i un KO Link en cas de que no. El significat d'èxit depèn del disseny intern de cada operació.
- Les operacions es poden unir en forma de seqüència a partir d'OK Link i KO Link.
- Una operació pot tenir qualsevol nombre de Transport Link de sortida per a enviar paràmetres a una altra operació. Els Transport Link no alteren l'execució de la cadena d'operacions.

Hi ha vuit tipus diferents d'unitats operacionals. A continuació se'n fa un breu resum.

- La *Create Unit* permet generar instàncies d'una entitat concreta.
- La *Delete Unit* permet eliminar una instància donat el seu identificador.
- La *Modify Unit* permet modificar els atributs d'una instància pertanyent a l'identificador del selector de la unitat.
- La *Connect Unit* permet relacionar dues instàncies de dues entitats relacionades.
- La *Disconnect Unit* permet eliminar relacions entre dues instàncies de dues entitats relacionades.
- La *Reconnect Unit* permet modificar la relació entre dues instàncies de dues entitats relacionades.
- La *Stored Procedure Unit* permet executar procediments de la base de dades-
- La *No Op Operation Unit* és una unitat sense lògica.

Per acabar, amb els links podem unir les pàgines i les unitats entre si. D'aquesta manera, tenim un punt d'inici a les accions des de les pàgines o simplement podem canviar d'una a l'altra. WebML distingeix entre dos tipus de links.

- Links contextuals
  Els links contextuals transporten informació entre les unitats a les que enllaça. Un link contextual pot ser qualsevol tipus de link: Transport Link, Normal Link, OK Link o KO Link.
- Links no contextuals
  Els links no contextuals són els que permeten lligar pàgines entre elles sense necessitat de traspassar-se informació. Un link no contextual no pot ser un Transport Link.

Hi ha dues maneres de passar informació entre les unitats a través dels links. Una és mitjançant el *coupling* o emparellament i permet enllaçar la informació que coneix la unitat origen amb els possibles paràmetres d'entrada de la unitat destí. L'altre és el *passing* que permet transportar la informació de forma que la unitat destí coneix aquesta informació per a poder-la passar a una tercera unitat si és necessari.

Un mateix link pot tenir variables en *coupling* i variables en *passing*, no són tipus excloents.

Dades (ER Model)

En aquest nivell tenim la representació del model de dades de l'aplicació. Consisteix en un model Entity – Relationship com ja explicarem més endavant.

5.1 WebRatio

WebRatio és una eina que permet fer models de WebML i generar automàticament el codi referent a l'especificació gràcies al XML i està desenvolupada per al mateix departament de la universitat *Politecnico di Milano* que va crear el WebML.

Aquesta eina és la que s'utilitzarà durant el projecte i ens permetrà fer el model de l'aplicació i comprovar-lo gràcies al codi generat.

6 Fòrum d'aprenentatge cooperatiu

6.1 Motivació

Amb la creació de les especialitats a l'enginyeria informàtica, els alumnes es centren en una sèrie d'assignatures i per tant el seu coneixement en un dels àmbits és major que en els altres.

Prenent això com a base, és lògic que cada alumne conegui millor tot el relacionat amb la seva especialitat però en canvi li manquin coneixements d'una altra branca que potser li facin falta en algun moment concret. Així doncs, l'objectiu de l'eina és la cooperació entre professors i alumnes per tal de respondre dubtes quan els coneixements d'una persona no són suficients.

Per a aquesta cooperació l'eina tindrà un fòrum públic on els usuaris podran obrir temes amb preguntes esperant resposta per part de gent més especialitzada però no es limitarà a aquest ús. A part de temes de preguntes respostes, es vol fomentar que els usuaris donin a conèixer noticies o publicacions sobre l'evolució de la informàtica a cada àmbit i sobre possibles conferències.

Així doncs, podem entendre el fòrum com a una matriu on en una dimensió tenim l'especialitat a la que correspon el tema i a l'altra el tipus de contingut (conferència, publicació o pregunta).

6.2 Descripció general

El fòrum el podem dividir de dues formes diferents segons el contingut dels fils o temes de discussió.

Per una banda, els podem separar segons l'especialitat de la que tracten. D'aquesta forma tenim cinc àrees diferents en aquesta dimensió.

- Computació
- Enginyeria de computadors
- Enginyeria del software
- Sistemes d'informació
- Tecnologies de la informació

Com a segona dimensió tenim el contingut del tema de discussió. En aquest cas diferenciem entre tres tipus diferents.

- Conferències o esdeveniments
- Publicacions
- Preguntes o dubtes

Cadascun dels temes de discussió haurà d'estar classificats en ambdues dimensions.

Tots els usuaris del fòrum pertanyen també a una especialitat. D'aquesta manera la resta d'usuaris poden saber el nivell de validesa d'una resposta.

A continuació es fa un petit resum de les funcionalitats que tindrà el fòrum. Més endavant estaran ampliades quan s'expliquin els casos d'ús de l'aplicació.

- **Registre d'usuaris**
  Els usuaris podran registrar-se amb un nom d'usuari, un e-mail i una especialitat. No podrà haver dues persones amb el mateix nom o e-mail.
- **Crear temes de discussió**
  Els usuaris podran crear temes de discussió de qualsevol tipus sempre que estigui ben classificat segons les normes del fòrum. Un tema necessitarà un títol i un contingut.
- **Modificar temes de discussió**
  L'autor dels temes de discussió podrà modificar el seu títol, contingut, especialitat o tipus de contingut.
- **Respondre a temes de discussió**
  Qualsevol usuari podrà respondre als temes de discussió del fòrum.
- **Modificar respostes de temes de discussió**
  L'autor d'una resposta podrà modificar el seu contingut.
- **Consultar el llistat de temes de discussió**
  Qualsevol usuari podrà consultar el llistat temes de discussió filtrats per contingut, especialitat o ambdós i veure un resum de la seva informació.
- **Consultar un tema de discussió**
  Qualsevol usuari podrà consultar els temes de discussió i veure la seva informació i la relacionada amb les seves respostes.

- **Poder fer temes de discussió preferits**
  Els temes de discussió preferits d'un usuaris són temes que l'usuari podrà veure sempre i accedir-hi a través de l'índex de l'aplicació. Aquests temes poden ser vistos per altres usuaris a través del perfil.
  Els mateixos temes de discussió preferits també es podran eliminar de la llista per l'usuari.
- **Perfil d'usuari**
  Cada usuari tindrà una sèrie de dades visibles per la resta de participants del fòrum. Entre aquestes dades, es podran veure el nom, l'e-mail, l'especialitat, els temes de discussió creats per l'usuari, les respostes que ha fet i els temes que té com a preferits.
- **Modificar dades d'usuari**
  L'usuari podrà modificar el seu e-mail en el cas de que vulgui canviar-lo.

6.3 Casos d'ús

Aquests són els casos d'ús que s'han decidit per a l'aplicació.

Els actors que distingim són els usuaris, que poden o no estar registrats.

6.3.1 Registre d'usuaris

Autor

Usuari no registrat.

Cas d'ús principal

1. L'actor selecciona crear un nou usuari.
2. El sistema sol·licita les dades per a la creació d'un usuari.
3. L'actor proveeix les dades per a la creació d'un usuari.
4. El sistema registra l'usuari i l'inicia la sessió.

Flux d'error

4. El sistema mostra un missatge d'error.
5. El flux continua des del punt 2.

6.3.2 Crear temes de discussió

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona crear un nou tema de discussió.
2. El sistema sol·licita les dades per a la creació d'un tema de discussió.
3. L'actor proveeix les dades per a la creació d'un tema de discussió.
4. El sistema crea el tema de discussió.

Flux d'error

4. El sistema mostra un missatge d'error.
5. El flux continua des del punt 2.

6.3.3 Modificar temes de discussió

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona modificar un tema de discussió propi.
2. El sistema mostra les dades actuals del tema de discussió.
3. L'actor modifica les dades pertinents.
4. El sistema guarda les modificacions del tema de discussió.

Flux d'error

4. El sistema mostra un missatge d'error.
5. El flux continua des del punt 2.

6.3.4 Respondre a temes de discussió

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona publicar una resposta a un tema de discussió existent.
2. El sistema sol·licita les dades per a la creació d'una resposta.
3. L'actor proveeix les dades per a la creació d'una resposta.
4. El sistema crea la resposta.

Flux d'error

4. El sistema mostra un missatge d'error.
5. El flux continua des del punt 2.

6.3.5 Modificar respostes de temes de discussió

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona modificar un resposta propia.
2. El sistema mostra les dades actuals de la resposta.
3. L'actor modifica les dades de la resposta.
4. El sistema guarda les modificacions de la resposta.

Flux d'error

4. Es mostra a l'usuari un missatge d'error.
5. El flux continua des del punt 2.

6.3.6 Consultar el llistat de temes de discussió

Autor

Usuari.

Cas d'ús principal

1. L'actor selecciona veure el llistat de temes de discussió.
2. El sistema mostra les dades actuals del filtratge.
3. El sistema mostra el resum d'informació dels temes seleccionats.
4. L'actor modifica les dades del filtratge.
5. El flux torna al punt 2.

6.3.7 Consultar un tema de discussió

Autor

Usuari.

Cas d'ús principal

1. L'actor selecciona veure la informació d'un tema de discussió.
2. El sistema mostra la informació relativa al tema seleccionat.

6.3.8 Poder fer temes de discussió preferits

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona fer un tema de discussió preferit.
2. El sistema afegeix el tema de discussió a la llista de preferits de l'actor.

Flux alternatiu

1. L'actor selecciona eliminar un tema de discussió preferit.
2. El sistema elimina el tema de discussió de la llista de preferits de l'actor.

6.3.9 Perfil d'usuari

Autor

Usuari.

Cas d'ús principal

1. L'actor selecciona visualitzar el perfil d'un usuari.
2. El sistema mostra les dades referents a l'usuari seleccionat.

6.3.10 Modificar dades d'usuari

Autor

Usuari registrat.

Precondició

L'usuari ha iniciat sessió al sistema.

Cas d'ús principal

1. L'actor selecciona modificar les seves dades.
2. El sistema mostra les dades actuals de l'usuari.
3. L'actor modifica les dades.
4. El sistema guarda les modificacions de les dades de l'usuari.

Flux d'error

4. Es mostra a l'usuari un missatge d'error.
5. El flux continua des del punt 2.

6.4 Model conceptual

Un cop vistos els casos d'ús que volem tractar a l'aplicació, s'extreu quina és la informació necessària que hem de mantenir al sistema i com està relacionada.

En el cas del **registre d'usuaris** necessitem poder guardar la informació amb la que s'ha registrat l'usuari. Aquesta és el nom, l'e-mail i la contrasenya i la representem en una classe User.

A part d'això, també tenim la informació de l'especialitat. Aquesta podria ser un atribut més de l'usuari però com volem restringir-les és preferible representar-ho en una nova classe (Specializaiton). Aquesta classe tindrà com atribut el nom de l'especialitat que serà a la vegada la clau primària.

Il·lustració 2: Classes i atributs pel registre d'usuaris

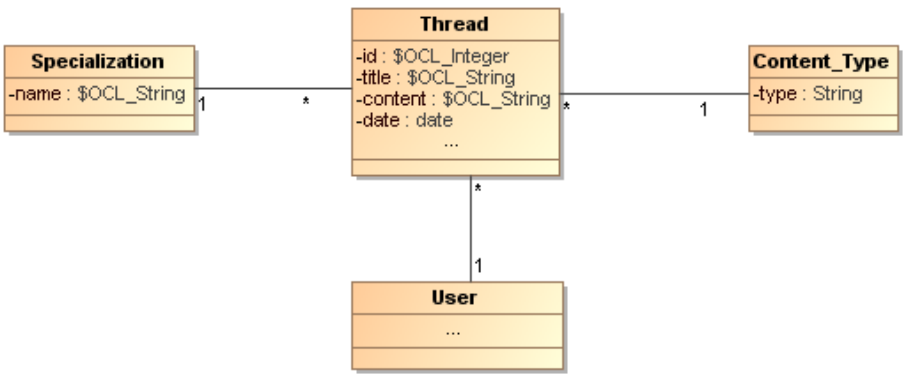
Per tal de **crear temes de discussió** haurem de tenir una nova classe per a representar la informació pròpia del tema (Thread). Un tema de discussió té un títol, un contingut i una data de creació. A més a més, com cap d'aquests atributs és únic d'un tema de discussió, s'afegeix un identificador per a poder diferenciar-los i tractar-los correctament a l'aplicació.

L'especialitat i el tipus de contingut podrien ser un altre atribut del tema de discussió però igual que restringim les especialitats i hem creat una classe que les representa, en el cas del tipus de contingut també és necessari.

Així doncs, crearíem una classe per al tipus de contingut (Content_Type) amb el tipus com a atribut i clau primària.

I ja per acabar amb la creació e temes de discussió, necessitem enllaçar les diferents classes.

- Té un únic autor (User) que pot haver fet més d'un tema de discussió.
- Té una única especialitat que pot ser el de més d'un tema de discussió.
- Té un únic tipus de contingut que pot ser el de més d'un tema de discussió.



Il·lustració 3: Classes i atributs per la creació de temes de discussió

Pels casos en que hem de **modificar temes de discussió** l'únic afegit nou seria guardar la data de modificació del tema de discussió per tal de poder-lo mostrar després de la modificació.

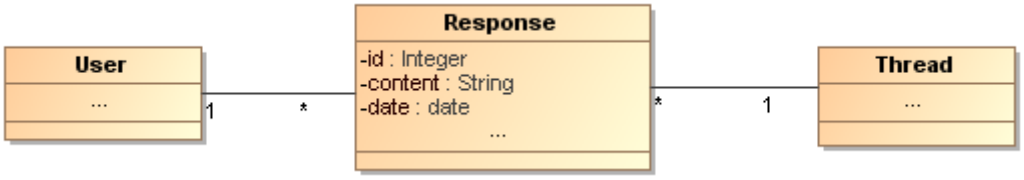
Il·lustració 4: Classes i atributs per la modificació de temes de discussió

Per **respondre a temes de discussió** necessitarem guardar d'alguna manera el contingut de la resposta i la data en la que s'ha fet. A més a més, una resposta sempre ha de tenir un autor i pertànyer a un tema de discussió.

Amb una nova classe Response, podem guardar el contingut i la data de la resposta però com cap d'aquests dos atributs són únics de les instàncies de resposta, s'afegeix un identificador per a poder diferenciar-los i utilitzar-lo de clau primària.

Les relacions amb l'autor i el tema de discussió serien:

- Té un únic autor (User) que pot haver publicat més d'una resposta.
- Pertany a un únic tema de discussió (Thread) que pot tenir més d'una resposta.



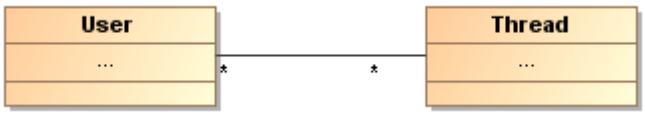
Il·lustració 5: Classes i atributs per a la creació de respostes

Igual que al modificar temes de discussió, **modificar respostes de temes de discussió** demana afegir un nou atribut a la classe Response amb la data de modificació.

Il·lustració 6: Classes i atributs per a la modificació de respostes

Els casos d'ús de **consulta del llistat de temes de discussió** i d'**un tema concret** mostraran informació relativa al tema en sí i no mostraran cap informació que no introduïm per altres vies. Només hem d'assegurar-nos de que un tema pertanyi a una especialitat i a un tipus de contingut per a fer el seu filtratge a la llista.

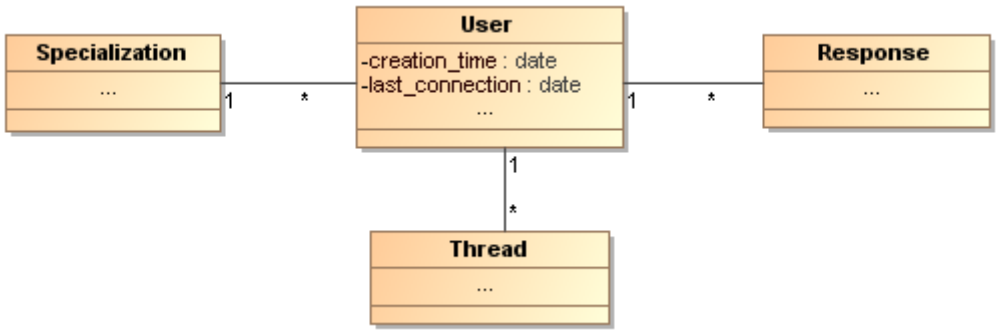
Per **poder fer temes de discussió preferits** tot el que necessitem és una nova relació entre els usuaris i els temes de discussió. En aquest cas, un usuari pot tenir tots els temes preferits que vulgui i més d'un usuari pot tenir un mateix tema de discussió com a preferit.



Il·lustració 7: Classes i atributs per a guardar preferits

El **perfil d'usuari** mostra tota la informació que pot ser important per a la resta d'usuaris. Aquesta conté moltes de les dades de la creació de l'usuari però a més es vol mostrar els temes de discussió que ha creat, les respostes que ha fet, els seus temes preferits, quan es va crear l'usuari i l'últim cop que va entrar al sistema.

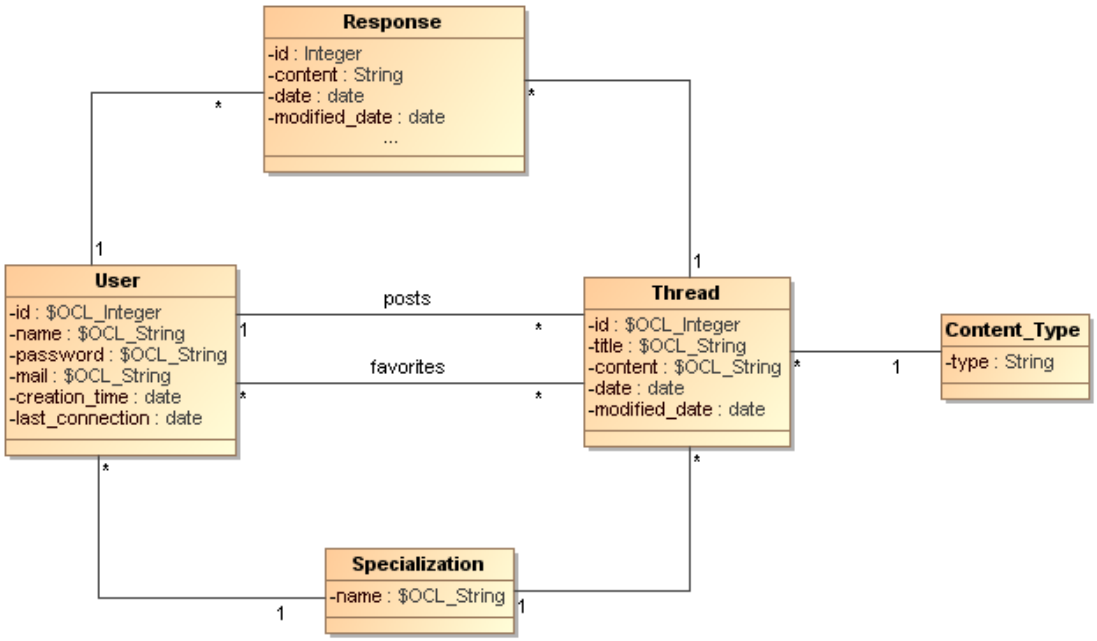
La informació relativa als temes creats, respostes i preferits ja està dins del sistema per casos d'ús anteriors però és necessari incorporar la data de creació i de l'última connexió a la classe User.



Il·lustració 8: Classes i atributs per al perfil d'usuari

Els casos d'ús de **login** i de **modificar les dades de l'usuari** no afecten al diagrama de classes.

Un cop dit tot això, es mostra a continuació el resultat de la unió de tots els requeriments.



Il·lustració 9: Resum de classes i atributs del model conceptual

Com es pot veure al diagrama final, als usuaris se'ls ha assignat un identificador numèric per a fer més fàcil el seu tractament a l'aplicació.

Aclariment:

En tots casos l'especialitat i el tipus de contingut podrien ser atributs enumerats. S'ha decidit implementar-los com a classes per semblança a la implementació en una base de dades.

6.5 UX Model

L'UX model descriu la informació que contenen les diferents finestres de l'aplicació i la navegació entre elles.

Aquest model serveix per a saber com representarem la informació de cara a l'usuari i les limitacions que aquest tindrà per a moure's d'un contingut cap a un altre.

A continuació s'explicarà quina és l'experiència que es vol que l'usuari tingui navegant pel web i el model resultant d'això.

6.5.1 Explicació

6.5.1.1 Login

Per començar, l'usuari ha de poder iniciar la sessió. Per a això, mostrarem al entrar al web un formulari on introduir el nom d'usuari i la contrasenya. Si són correctes les dades, redirigirem a l'usuari a la pàgina d'índex de l'aplicació i en cas de ser incorrectes romandrem a la pàgina amb un missatge d'error.

6.5.1.2 Registre

En el cas de que l'usuari no estigui registrat, aquest ha de poder-ho fer mitjançant un formulari amb el nom d'usuari, la contrasenya, el seu e-mail i la seva especialitat. Aquest formulari estarà en una altra pàgina i igual que l'anterior, en cas d'èxit anirà a la pàgina d'índex i en cas d'error romandrà a la mateixa pàgina amb el missatge.

Des d'aquesta pàgina es pot tornar a la pàgina d'inici.

6.5.1.3 Índex

L'índex de l'aplicació serà accessible des de qualsevol altra pàgina que no sigui la de registre o la d'entrada a l'aplicació. Aquesta mostrarà informació sobre els temes de discussió del sistema i els temes preferits de l'usuari. Per a cada tema mostrarà el títol, la data de creació, el nombre de respostes, el tipus de contingut, l'especialitat i el nom de l'autor.

Des de cada un dels resums dels temes és possible anar a veure el tema concret o al perfil de l'usuari que l'ha creat. Si el tema és preferit, es pot eliminar de la llista de preferits de l'usuari. També és possible accedir a la pàgina de creació d'un nou tema de discussió o a la modificació de les dades de l'usuari actual des d'aquesta.

Els temes generals podran ser filtrats a partir d'un formulari segons tipus de contingut i especialitat. En cas de no escollir cap concret en un filtre, es tindrà en compte com a si fossin tots.

Tant uns temes com els altres es mostraran amb paginació en finestres de deu.

6.5.1.4 Crear tema de discussió

La creació d'un tema de discussió es fa mitjançant un formulari consistent en una especialitat i un tipus de contingut a escollir entre els possibles, un títol i un contingut. L'usuari ha d'haver iniciat sessió al sistema per poder crear un nou tema. Un cop creat el tema correctament es mostrarà la pàgina amb la informació completa del tema de discussió. En cas de qualsevol error, es tornarà a mostrar la pàgina del formulari amb la informació de l'error.

6.5.1.5 Informació d'un tema

Ja sigui des de l'índex o des del perfil de l'usuari es pot accedir a veure la informació completa d'un tema de discussió igual que després de la creació d'aquest. Aquesta pàgina mostra el títol, el contingut, l'especialitat i el tipus de contingut del tema a més del nom de l'autor del tema i la data de publicació, així com el contingut, la data i l'autor de totes les respostes que s'hagin fet referents a aquell tema de discussió.

Es pot anar tant a la modificació del tema de discussió com de cada una de les respostes des d'aquesta pàgina. En cas de que el tema o alguna resposta ja sigui modificada, es mostrarà també la data de modificació.

Per acabar, també és possible afegir el tema a la llista de preferits de l'usuari o eliminar-lo d'aquesta.

6.5.1.6 Afegir/eliminar un tema de discussió preferit

Des de la pàgina d'informació dels temes de discussió, l'usuari pot afegir el tema a la llista de preferits per tal de que es mostri a la pàgina d'índex.

Des de l'índex o des de la pàgina d'informació dels temes de discussió, l'usuari pot eliminar un tema de la llista de preferits.

En cas d'èxit es mostrarà la pàgina d'informació del tema de discussió.

6.5.1.7 Modificació del tema de discussió

La pàgina de modificació del tema de discussió és igual que la de creació però el formulari estarà omplert al començament amb la informació actual del tema. En el cas de buidar un dels camps, aquest camp no es modificarà. Si s'efectua qualsevol error durant la modificació es romandrà a la mateixa pàgina amb la informació de l'error i en cas d'èxit es tornarà a la pàgina d'informació del tema de discussió que s'ha modificat.

6.5.1.8 Modificació d'una resposta

La modificació d'una resposta es farà des d'un formulari on es mostrarà el contingut actual de la resposta. En cas de deixar en blanc el camp del formulari, no s'efectuarà cap modificació. Si es produeix algun error al fer la modificació, es romandrà a la mateixa pàgina mostrant la informació de l'error corresponent i si es produeix amb èxit es tornarà a la pàgina d'informació del tema al que pertany la resposta.

6.5.1.9 Perfil d'usuari

El perfil d'un usuari mostra el seu nom, e-mail, data de creació i data d'última connexió així com un llistat del resum de la informació dels temes de discussió que ha publicat, els temes on ha respost junt amb la data amb la que va publicar la resposta i els temes que té com a preferits.

Des d'aquesta pàgina es podrà accedir a la pàgina d'informació de tots els temes de discussió que es mostren i a la modificació de les dades de l'usuari actual del sistema.

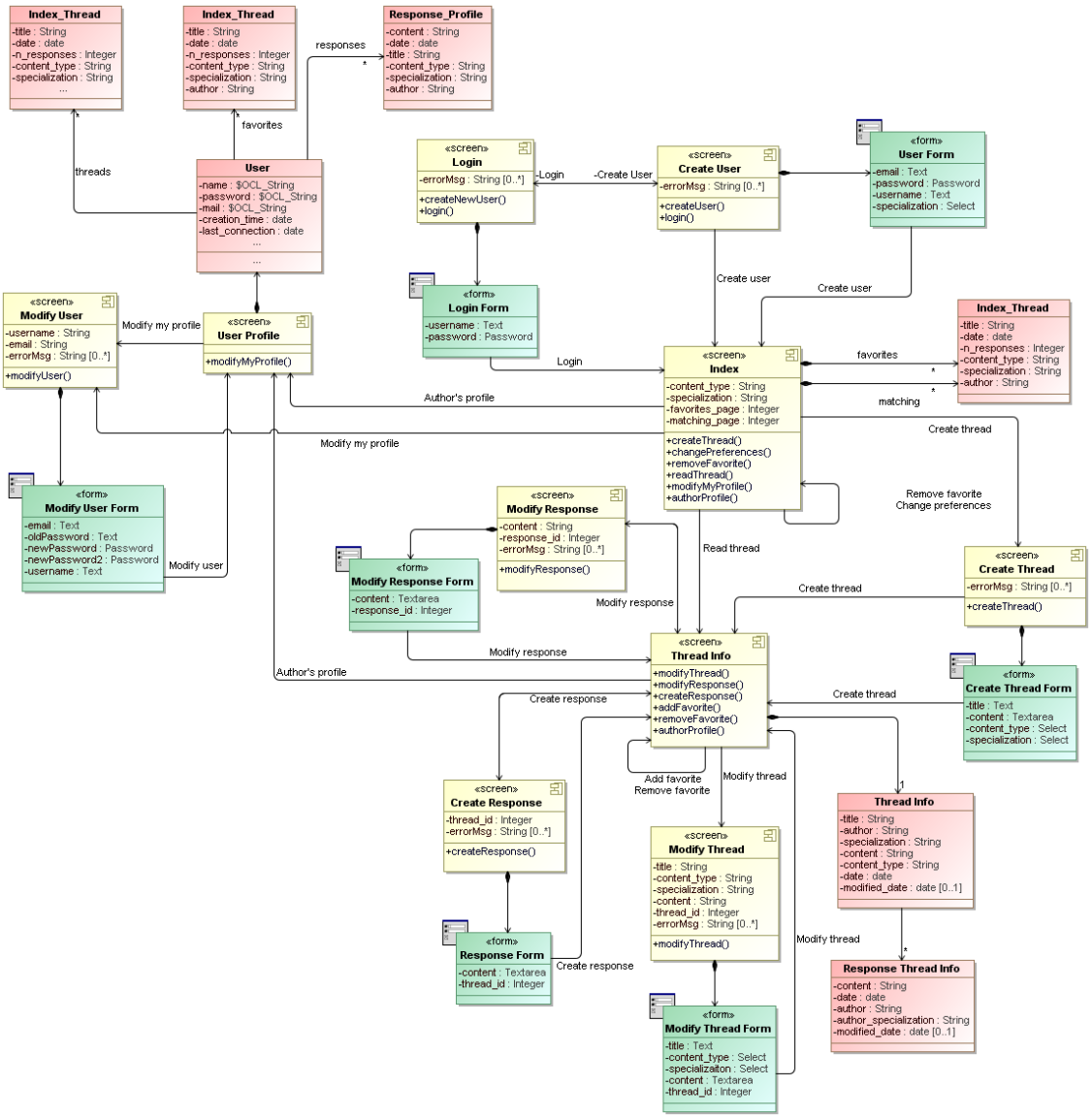
Tota la informació referent als temes de discussió es mostrarà amb paginació amb finestres de deu.

6.5.1.10 Modificació de les dades de l'usuari

A partir d'un formulari es podrà modificar l'e-mail o la contrasenya de l'usuari actual del sistema. El formulari estarà omplert amb l'e-mail actual a més del nom d'usuari tot i que aquest no es pugui modificar. Es podrà anar al perfil de l'usuari actual des d'aquesta pàgina si es decideix no guardar els canvis.

Per tal de canviar la contrasenya, l'usuari haurà d'introduir la contrasenya actual així com dues vegades la nova contrasenya. Per a canviar l'e-mail no caldrà cap contrasenya.

6.5.2 Model resultant



Il·lustració 10: UX Model

Tal i com s'ha explicat, totes les pàgines excepte Login i Create User poden anar cap a Index o a Logout.

6.6 Diagrama intern de l'aplicació (WAE)

Per tal d'especificar correctament el disseny web amb UML s'ha d'utilitzar la *Web Application Extension* o WAE. Això ens permet representar en UML les pàgines i altres elements significants del disseny intern de la presentació sistema. Aquesta extensió està proposada per Jim Conallen al llibre "Building Web Applications with UML".

Els estereotips que s'han utilitzat en aquest projecte són:

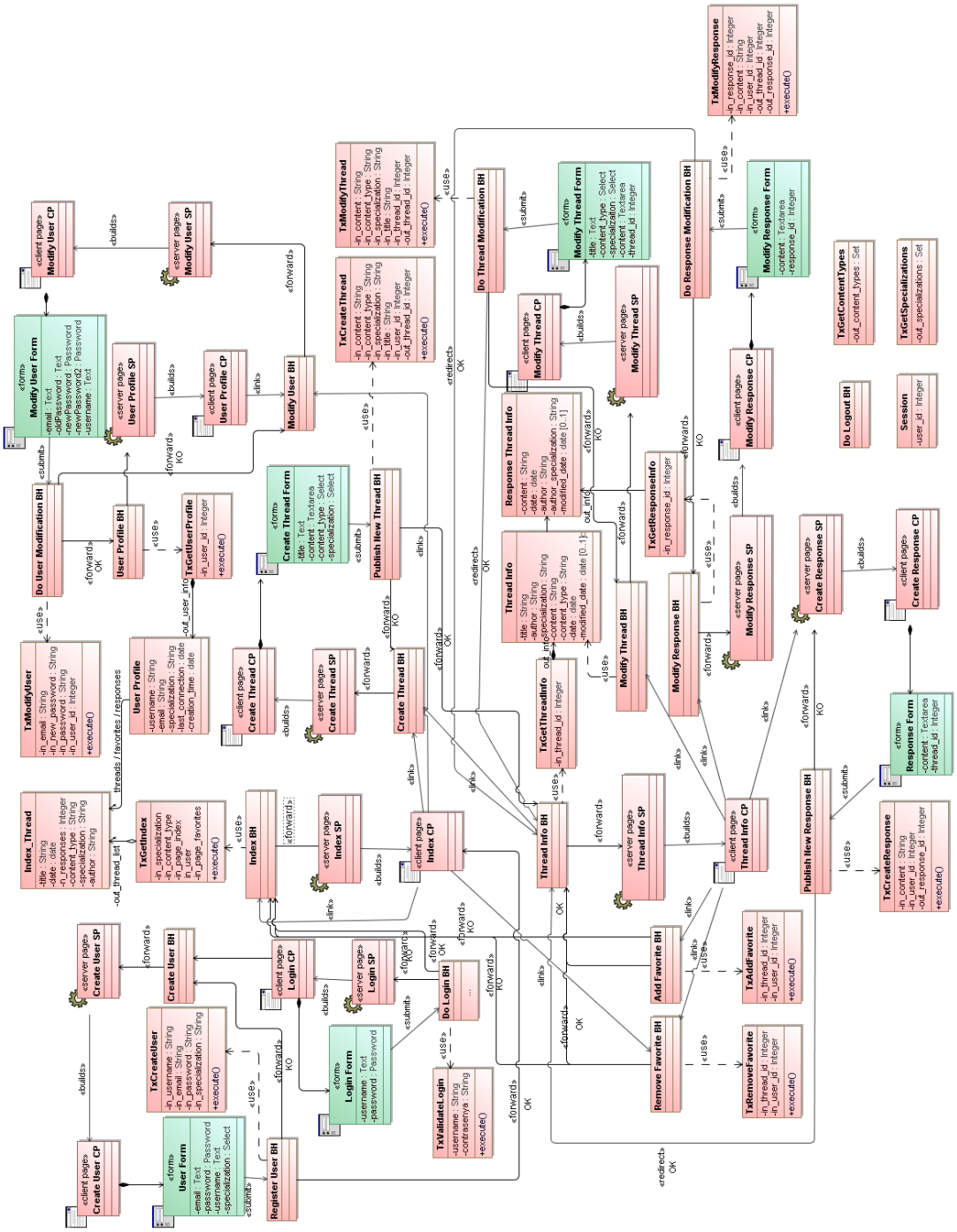
- Classes
  - Client Page
    Instància representant d'una pàgina HTML. Conté com a operacions els possibles links des de la pàgina.
  - Server Page
    Crea i munta les *client page* amb la informació que rep.
  - Business helper
    Encarregat d'interactuar amb el domini del sistema per a extreure la informació que necessita la *server page*.
  - Form
    Representa un formulari i formarà part sempre d'una *client page*. Els seus atributs són els camps del formulari.
- Associacions
  - Link
    Enllaça una *client page* amb una *server page* o *business helper*. Pot portar informació.
  - Build
    Munta una *client page* a partir d'una *server page*.
  - Submit
    Des d'un formulari indica l'acció cap a una *server page* o un *business helper*.
  - Redirect
    Fa canviar al navegador de direcció. Pot anar d'una *client page*, una *server page* o un b*usiness helper* a una altra d'aquestes tres.
  - Forward
    Indica un enviament d'informació i canvi de context des d'una *Server Page* o un *business helper* a un altre d'aquests dos.

A més a més, utilitzarem també una classe Session on guardarem la informació corresponent a la sessió actual de l'usuari i les transaccions de la capa de domini que definirem en aquest punt.

L'estructura de classes ha vingut donada per l'aplicació del patró *Service to Worker* proposat al llibre de "Core 2JEE Patterns: Best Practices and Design Strategies". Això ens fa separar la lògica de la *server page* i posar-la als *business helper*.

A continuació es mostrarà l'esquema general i s'explicarà per a cada *server page*, *business helper* i transacció quina lògica efectuen.

6.6.1 Model



Il·lustració 11: Diagrama intern de l'aplicació

A Do Login BH hi ha relació des de qualsevol *Client Page* excepte Login CP i Create User CP.

TxGetSpecialization s'utilitza des de Create Thread BH, Create User BH, Index BH, Modify Thread BH, i Modify User BH.

TxGetContentTypes s'utilitza des de Create Thread BH, Index BH i Modify Thread BH.

Session és utilitzada des de tots els *business helper* excepte Create User BH i Thread Info BH.

6.6.2 Explicacions

6.6.2.1 Transaccions

TxAddFavorite

A partir d'un identificador d'usuari i un identificador d'un tema de discussió, afegeix el tema de discussió a la llista de temes de discussió preferits per l'usuari.

Genera una excepció en el cas de no existir l'usuari o el tema de discussió.

TxCreateResponse

A partir d'un contingut, un identificador de tema de discussió i un identificador d'usuari crea una nova resposta a un tema de discussió al sistema.

Genera una excepció en el cas de no existir l'usuari o el tema de discussió o si qualsevol dels camps és *null*.

TxCreateThread

A partir d'un títol, un contingut, un tipus de contingut, una especialitat i un identificador d'usuari crea un nou tema de discussió al sistema.

Genera una excepció en el cas de no existir l'usuari o si qualsevol dels camps és *null*.

TxCreateUser

A partir d'un nom d'usuari, un e-mail, una contrasenya i una especialitat crea un nou usuari al sistema.

Genera una excepció en el cas d'existir un usuari al sistema amb el mateix nom d'usuari o e-mail, si qualsevol dels camps és *null* o l'especialitat no és una de les registrades al sistema.

TxGetContentTypes

Retorna el llistat de noms dels tipus de contingut del sistema.

TxGetIndex

A partir del nom d'un tipus de contingut, el nom d'una especialitat, el número de pàgina corresponent als temes filtrats, el número de pàgina corresponent als temes preferits i l'identificador d'un usuari retorna la informació dels deu temes corresponents a la pàgina dels temes filtrats aplicant el filtre del tipus de contingut i de l'especialitat sobre tots els temes de discussió del sistema i dels deu temes preferits corresponents a la pàgina dels temes preferits de l'usuari.

Per a cada tema de discussió es retorna el títol, la data de creació, el nombre de respostes del tema de discussió, el nom del seu tipus de contingut, el nom de la seva especialitat i el nom de l'autor.

TxGetSpecializations

Retorna el llistat de noms de les especialitats del sistema.

TxGetThreadInfo

A partir d'un identificador de tema de discussió, retorna la informació referent al tema de discussió i a les seves respostes.

Per al tema de discussió retorna el títol, el nom de l'autor, el nom de l'especialitat, el nom del tipus de contingut, el contingut, la data de creació i ,si hi ha, la data de modificació del tema de discussió.

Per a cada una de les respostes referents al tema de discussió retorna el contingut, la data, el nom de l'autor, l'especialitat de l'autor i, si hi ha, la data de modificació de la resposta.

TxGetUserProfile

A partir d'un identificador d'usuari, retorna la informació relativa a l'usuari, als temes de discussió que ha creat, a les respostes que ha publicat i als seus temes de discussió preferits.

De l'usuari retorna el seu nom d'usuari, el seu e-mail, el nom de la seva especialitat, la data de creació i la data d'última connexió a l'aplicació.

Per a cada tema de discussió creat per l'usuari retorna el títol, la data de creació, el nombre de respostes del tema de discussió, el nom del seu tipus de contingut i el nom de la seva especialitat.

Per a cada resposta feta per l'usuari retorna el seu contingut i del tema de discussió al que pertany, el títol, la data de creació, el nom del seu tipus de contingut, el nom de la seva especialitat i el nom de l'autor.

Per a cada tema preferit de l'usuari retorna el títol, la data de creació, el nombre de respostes del tema de discussió, el nom del seu tipus de contingut, el nom de la seva especialitat i el nom de l'autor.

TxModifyResponse

A partir d'un identificador de resposta i un contingut, substitueix el contingut actual de la resposta pel nou contingut en el cas de que aquest no sigui buit i actualitza la seva data de modificació.

Retorna l'identificador del tema de discussió al que pertany la resposta.

Genera una excepció en el cas de no existir la resposta.

TxModifyThread

A partir d'un identificador de tema de discussió, un nom de tipus de contingut, un nom d'especialitat, un títol i un contingut, substitueix la informació actual del tema de discussió per a tota aquella que hi hagi d'entrada i no sigui buida i actualitza la seva data de modificació.

Retorna l'identificador del tema de discussió que s'ha modificat.

Genera una excepció en el cas de no existir el tema de discussió o si el nom de l'especialitat o el tipus de contingut no és un dels existents al sistema.

TxModifyUser

A partir d'un identificador d'usuari, un e-mail, una contrasenya actual i una contrasenya nova, modifica l'e-mail de l'usuari si aquest no és buit i modifica la contrasenya actual en el cas de que l'actual passada per paràmetre sigui igual a la registrada al sistema i la nova no sigui buida.

Genera una excepció en cas de no existir l'usuari o si la contrasenya actual no és buida i no coincideix amb la registrada al sistema.

TxRemoveFavorite

A partir d'un identificador d'usuari i un identificador d'un tema de discussió, elimina el tema de discussió a la llista de temes de discussió preferits per l'usuari.

Genera una excepció en el cas de no existir l'usuari o el tema de discussió.

TxValidateLogin

A partir d'un nom d'usuari i una contrasenya comprova que la contrasenya sigui igual que la registrada al sistema per al nom d'usuari.

Retorna l'identificador de l'usuari en cas de que coincideixi i retorna *null* si no.

6.6.2.2 Business Helper

Login BH

Consulta si la informació per a entrar al sistema és correcte a partir de la transacció TxValidateLogin.

En cas de ser-ho guarda l'usuari en sessió i redirigeix a Index BH.

En cas de ser incorrecte crea el missatge d'error i segueix l'execució amb la informació a Login SP.

Create User BH

Extreu la informació necessària sobre l'especialitat per a generar el formulari de creació d'usuari a partir de la transacció TxGetSpecialization.

Envia aquesta informació i la de possibles errors que hagi rebut a Create User SP on continua l'execució.

Register User BH

Utilitza la transacció TxCreateUser per a crear un nou usuari al sistema.

Si es produeix amb èxit el registre, guarda l'usuari a sessió i segueix l'execució a Index BH.

En cas de capturar alguna excepció s'envia la informació de l'error a Create User BH i prossegueix l'execució des d'allà.

Index BH

Extreu la informació sobre les possibles especialitats i tipus de continguts sobre les que es poden filtrar els temes de discussió a partir de les transaccions TxGetSpecializations i TxGetContentTypes.

També extreu la informació sobre els temes de discussió segons el filtratge (tots per defecte) i els favorits de l'usuari de la sessió amb la transacció TxGetIndex.

Envia aquesta informació a Index SP on continua l'execució.

Create Thread BH

Extreu la informació sobre les possibles especialitats i tipus de continguts per a la creació d'un nou tema de discussió a partir de les transaccions TxGetSpecializations i TxGetContentTypes.

Envia aquesta informació i la de possibles errors de creació que hagi rebut a Create Thread SP on continua l'execució.

Publish New Thread BH

Utilitza la transacció TxCreateThread per a crear una nova instància de tema de discussió amb l'usuari de la sessió actual com a autor.

Si es produeix amb èxit la creació s'envia l'identificador a Thread Info BH on continua l'execució.

En el cas de capturar alguna excepció s'envia la informació de l'error a Create Thread BH i prossegueix l'execució des d'allà.

Thread Info BH

Utilitza la transacció TxGetThreadInfo per a extreure la informació sobre el tema de discussió que es vulgui mostrar.

S'envia aquesta informació a Thread Info SP on continua l'execució.

En cas de capturar l'excepció de que no existeix el tema de discussió, es redirigeix el flux cap a Index BH.

Publish New Response BH

Utilitza la transacció TxCreateResponse per a crear una nova instància de resposta a un tema de discussió amb l'usuari de la sessió actual com a autor.

Si es produeix amb èxit la creació s'envia l'identificador del tema de discussió al que pertany a Thread Info BH.

En cas de capturar l'excepció de que no existeix el tema de discussió, es redirigeix el flux a Index BH.

En cas de capturar qualsevol altra excepció, s'envia la informació de l'error a Create Response SP.

Add Favorite BH

Utilitza la transacció TxAddFavorite per a afegir un tema de discussió als preferits de l'usuari de la sessió en curs.

En cas de capturar l'excepció de que no existeix el tema de discussió l'execució continua a Index BH.

En cas de capturar l'excepció de que no existeix l'usuari en sessió l'execució continua a Login BH.

Remove Favorite BH

Utilitza la transacció TxRemoveFavorite per a eliminar un tema de discussió dels preferits de l'usuari de la sessió en curs.

En cas de capturar l'excepció de que no existeix el tema de discussió l'execució continua a Index BH.

En cas de capturar l'excepció de que no existeix l'usuari en sessió l'execució continua a Login BH.

Modify Thread BH

Utilitza la transacció TxGetThreadInfo per a extreure la informació actual del tema de discussió que es vol modificar i TxGetContentTypes i TxGetSpecializations per a agafar les llistes de possibles tipus de contingut i especialitats per al tema.

En el cas de que l'usuari en sessió no sigui l'autor del tema de discussió que es vol modificar, es redirigeix el flux a Thread Info BH amb l'identificador del tema de discussió.

En cas d'èxit envia aquesta informació i la de possibles errors de creació que hagi rebut a Modify Thread SP on continua l'execució.

En cas de capturar l'excepció de que el tema de discussió no existeix, l'execució continua a Index BH.

Do Thread Modification BH

Utilitza la transacció TxModifyThread per a modificar les dades actuals d'un tema de discussió.

Si es produeix amb èxit la modificació s'envia l'identificador del tema de discussió modificat a Thread Info BH on prossegueix l'execució.

En cas de capturar l'excepció de que el tema de discussió no existeix, l'execució continua a Index BH.

En cas de capturar qualsevol altra excepció, s'envia la informació de l'error a Modify Thread BH.

Modify Response BH

Utilitza la transacció TxGetResponseInfo per a extreure les dades de la resposta que volem modificar.

Envia aquesta informació i la de possibles errors de creació que hagi rebut a Modify Response SP on continua l'execució.

En cas de que l'usuari en sessió no sigui l'autor de la resposta, envia l'identificador del tema de discussió al que pertany la resposta a Thread Info BH.

En cas de capturar l'excepció de que no existeix la resposta o no tingui cap autor relacionat, l'execució continua a Index BH.

Do Response Modificaiton BH

Utilitza la transacció TxModifyResponse per a modificar les dades actuals d'una resposta.

Si es produeix amb èxit la modificació s'envia l'identificador del tema de discussió al que pertany la resposta modificada a Thread Info BH on prossegueix l'execució.

En cas de capturar qualsevol excepció, s'envia la informació de l'error a Modify Response BH.

Modify User BH

Utilitza la transacció TxGetUserInfo per a extreure les dades actuals de l'usuari en sessió.

Envia aquesta informació i la de possibles errors de creació que hagi rebut a Modify User SP on continua l'execució.

En cas de capturar l'excepció de que no existeix l'usuari o no haver cap usuari en sessió, l'execució continua a Login SP.

Do User Modification BH

Utilitza la transacció TxModifyUser per a modificar les dades actuals d'una resposta.

Si es produeix amb èxit la modificació s'envia l'identificador de l'usuari modificat a User Profile BH on prossegueix l'execució.

En cas de capturar qualsevol excepció, s'envia la informació de l'error a Modify User BH.

Logout BH

Neteja la informació de la sessió i segueix l'execució a Login BH.

6.6.2.3 Server Pages

Create Response SP

Genera la pàgina amb el formulari per a crear una resposta a un tema de discussió i si ha rebut un missatge d'error per mostrar, el mostra.

Create Thread SP

Genera la pàgina amb el formulari per a crear un tema de discussió i si ha rebut un missatge d'error per mostrar, el mostra.

Create User SP

Genera la pàgina amb el formulari per a crear un usuari i si ha rebut un missatge d'error per mostrar, el mostra.

Index SP

Genera la pàgina amb el formulari per al filtre dels temes de discussió mostrats amb el valor actual seleccionat.

Mostra tots els temes de discussió rebuts des d'Index BH que són tots els temes filtrats de la pàgina actual a més dels preferits de l'usuari en sessió.

Login SP

Genera la pàgina amb el formulari d'entrada al sistema per a un usuari i si ha rebut un missatge d'error per mostrar, el mostra.

És la primera classe en executar-se al iniciar l'aplicació.

Modify Response SP

Genera la pàgina amb el formulari de modificació d'una resposta amb la informació d'aquesta que ha rebut des de Modify Repsonse BH i si ha rebut un missatge d'error per mostrar, el mostra.

Modify Thread SP

Genera la pàgina amb el formulari de modificació d'un tema de discussió amb la informació d'aquest que ha rebut des de Modify Thread BH i si ha rebut un missatge d'error per mostrar, el mostra.

Modify User SP

Genera la pàgina amb el formulari de modificació d'un usuari amb la informació d'aquest que ha rebut des de Modify User BH i si ha rebut un missatge d'error per mostrar, el mostra.

Thread Info SP

Genera la pàgina amb la informació completa d'un tema de discussió i les seves respostes amb la informació que rep de Thread Info BH.

User Profile SP

Genera la pàgina amb la informació completa d'un usuari i les seves respostes amb la informació que rep de User Profile BH.

7 D'UML a WebML – Traducció

Un cop feta l'especificació, toca traduir dels models i diagrames UML als de WebML.

S'ha optat per a una explicació *bottom-up* des de la base de dades fins a la navegació entre les diferents capes.

El que es podrà observar per a cada un dels passos que s'han seguit és pas a pas com s'ha anat traduint cada funcionalitat especificada a l'UML en cada un dels nivells de l'aplicació.

Començarem amb la base de dades on explicarem el model Entity – Relationship en el que es basa WebML i la manera amb la que s'ha adaptat el diagrama de classes original d'UML.

Després traduïm la unitat més interna que vindrien a ser les transaccions, que són les que interactuen directament amb la base de dades en els diagrames UML. S'ha explicat quines decisions estructurals s'han pres per a l'hora de traduir i s'han explicat pas a pas els passos per a generar els diagrames de WebML.

A nivell de l'esquema intern de l'aplicació, s'ha separat entre les pàgines i els *business helper*. Com s'explica més endavant, s'ha decidit separar alguna lògica d'extracció d'informació de dels *bussiness helper* i posar-la directament a la pàgina o el que podria ser la *server page*. De la mateixa forma que amb les transaccions, els *business helper* i les pàgines estan explicades pas a pas.

Un cop tenim tots els elements definits només quedava unir-los i això és el que s'explica al subpunt d'unió de les diferents capes. S'ha separat en dues parts per tal de que s'entengui millor. Primer com unim els resultats dels *business helper* amb les pàgines i després com unim les pàgines amb els *business helper* per a tal d'inicialitzar les funcionalitats i entre elles mateixes per a la correcta navegació.

En tot moment l'ordre de les explicacions està fet segons l'ordre alfabètic de les classes a traduir.

Per acabar amb la traducció, s'han exposat alguns dels problemes més recurrents trobats durant el projecte i que poden estalviar molt de temps en projectes futurs.

7.1 Model de dades

7.1.1 Enhanced Entity – Relationship Model (EER)

El Data Model de WebRatio és la representació de l'estructura de la capa de dades de l'aplicació. Aquesta representació es fa mitjançant un diagrama Enhanced Entity-Relationship (EER). En aquest punt s'explicarà una forma de traduir el diagrama de classes d'UML al EER per tal de poder seguir treballant més tard.

El primer que farem serà entrar en els conceptes i elements dels diagrames Entity-Relationship per a poder entendre millor el que s'ha de fer.

Hi ha quatre elements essencials en el modelatge ER:

- **Entitats** o classes corresponents als objectes del món real distingibles. En el cas dels EER, de forma diferent als ER, podem utilitzar la generalització i l'especialització.
- **Atributs** que defineixen l'estructura de les entitats. Dues instàncies de la mateixa entitat es diferencien pels seus atributs.
  Per assegurar això cal trobar un atribut clau que sigui diferent per a cada possible instància com podria ser el DNI per a les persones.
  Es poden afegir uns atributs anomenats **propietats** que permeten donar un valor fixe a totes les instàncies d'una mateixa entitat.
- **Relacions** entre diferents entitats per tal de modelar la interacció entre diferents objectes. Igual que en el cas de l'UML, que hi hagi una relació entre dues entitats no vol dir que totes les instàncies d'una entitat necessàriament hagin d'estar associades a una de l'altre entitat (Excepte el cas de * - 1).
- **Atributs de les relacions** que identifiquen informació necessària en el moment de la interacció entre dues instàncies.

7.1.1.1 Derivació

Tot i que no s'ha utilitzat en el projecte, també es poden utilitzar atributs derivats.

Segons el tipus d'inferència que fem WebRatio distingeix en quatre tipus diferents d'atributs derivats.

- **Atributs constants**
  D'aquesta forma podem donar un valor fixe a un atribut d'una entitat per a totes les seves instàncies.
- **Atributs importats**
  La importació d'atributs serveix per a copiar valors d'atributs d'altres entitats a través de relacions navegables de forma que l'entitat destí sempre té cardinalitat 1.
  En aquest cas distingim dos tipus d'importacions:
  - Simples
    S'entenen com a simples aquells atributs que copiem d'una entitat associada a l'entitat d'origen. Es demanarà a l'entitat original cada cop que es necessiti el valor.
  - Complexes
    S'entenen com a complexes els atributs que importem a través de vistes creades ja que s'afegeixen condicions.

- **Atributs calculats**
  A partir de l'agregació de valors podem obtenir la derivació per a l'atribut. Estan permesa la suma, mitjana, mínim i màxim. Utilitzat en relacions 1 a N.
- **Atributs de *mapping* manual**
  S'utilitza quan no podem fer la derivació de cap de les formes anteriors i ens permet escriure directament el codi SQL per a crear la vista a la base de dades i poder fer el *mapping* a l'atribut.

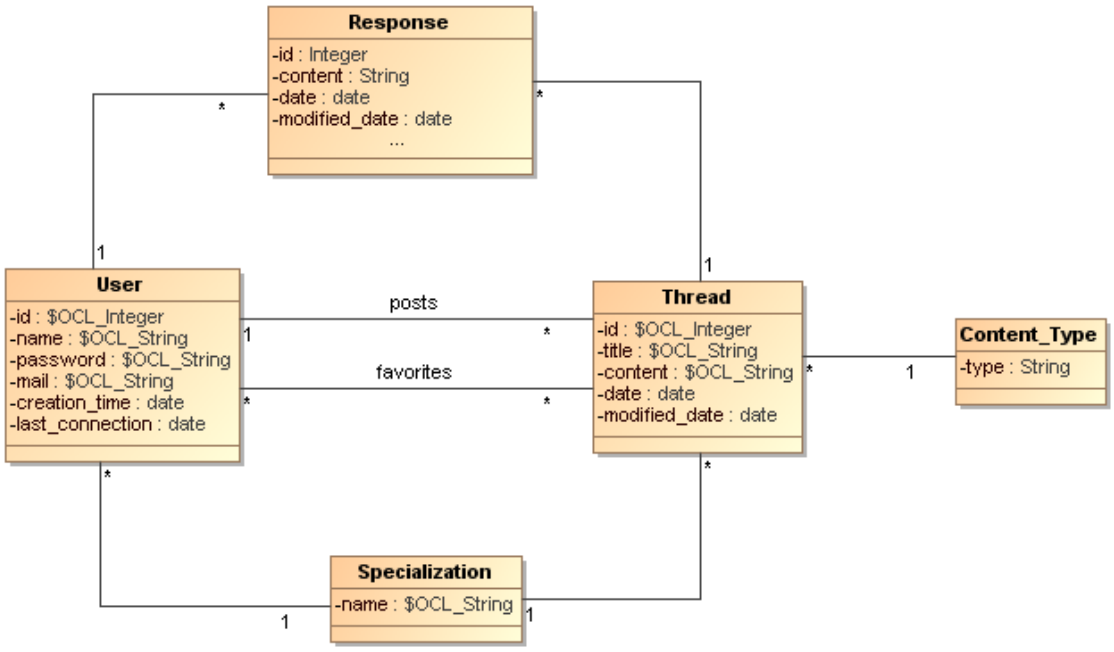
A més a més de tot això, també és possible representar més restriccions d'integritat gràcies a la derivació d'entitats i associacions.

A través de la **derivació d'entitats** podem definir restriccions a la pertinença a les subclasses d'una entitat segons condicions d'aquesta. Les condicions no estan limitades als atributs de la superclasse.

La **derivació d'associacions** ens permet modelar el comportament d'associacions de tipus *subset* o associacions on totes les parelles han d'estar també relacionades en una altra associació, 'concatenació' i restriccions a les entitats relacionades. Per exemple, podríem obligar en una relació que l'entitat d'origen tingui un atribut major que un atribut de l'entitat de destí.

7.1.2 Traducció

A continuació mostrarem el diagrama original de classes en UML i el *data model* resultant a WebRatio que s'ha utilitzat per a l'aplicació.

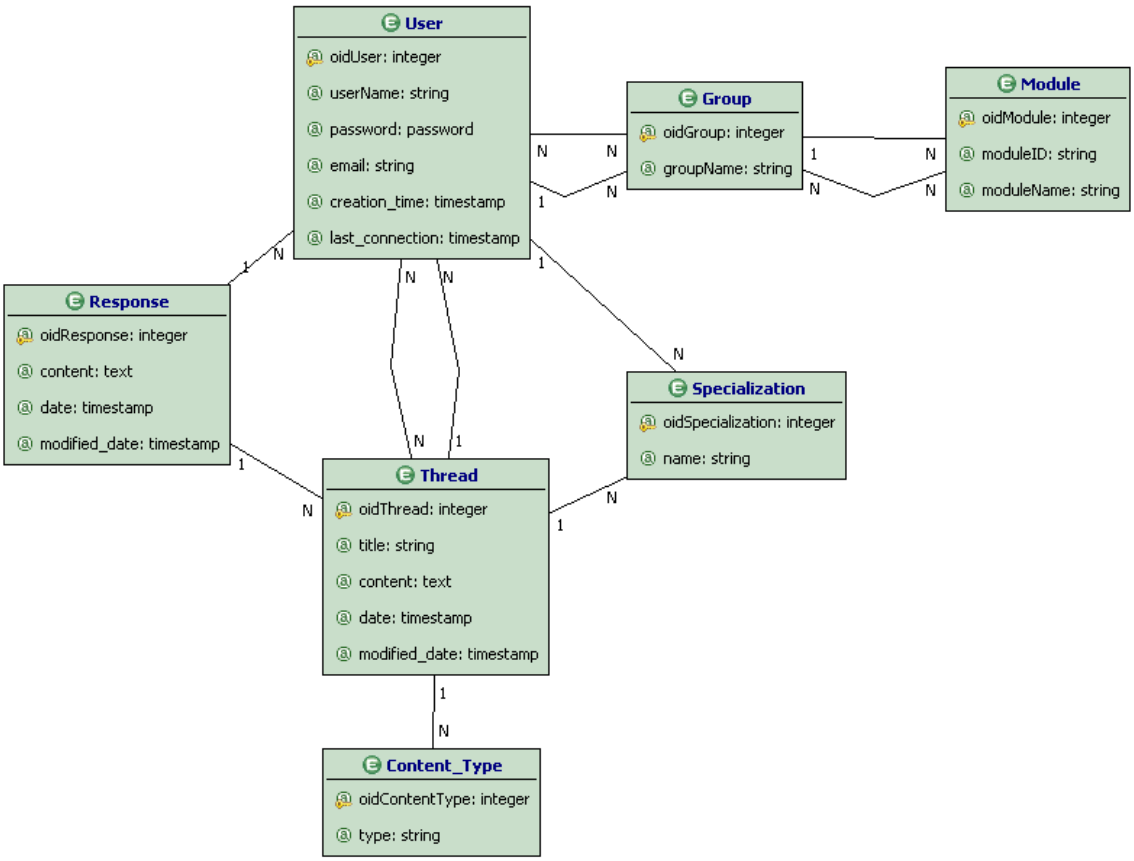


Il·lustració 12: Classes i atributs del model conceptual (UML)

Quan creem un projecte a WebRatio, el model de dades ja inclou una mínima estructura composada per les classes User, Group i Module. Aquestes classes, utilitzades correctament, permeten gestionar més d'un grup de persones dins de l'aplicació, separar-los per zona i

restringir l'accés a certes zones. Tot i això, en el projecte no s'ha utilitzat aquesta funcionalitat perquè només hi ha un tipus d'usuari i no s'ha cregut necessari. És per a aquest motiu que tenim les classes Group i Module dins del diagrama tot i que no ens facin falta.

També podem fixar-nos en que s'han afegit uns identificadors a algunes classes que no en tenien. WebRatio no permet posar valor *unique* i obliga a totes les instàncies a tenir com a clau primària el seu identificador intern. Per això, al diagrama no podrem representar quins valors formen la clau real d'una entitat i haurem de tractar-ho sempre des de domini a l'hora de crear noves instàncies.



Il·lustració 13: Classes i atributs del model conceptual (WebML)

Els identificadors tenen tots nom "oid" per defecte però és molt més còmodes canviar-los el nom segons l'entitat a la que representen per a no tenir problemes a l'hora d'enllaçar paràmetres.

Per últim, veure que sembla que les relacions siguin les contraries que al model de dades d'UML. Això no és cosa de l'EER sinó de com WebRatio mostra el diagrama.

A continuació es mostrarà la informació sobre els links de WebRatio. Per a totes es diu l'origen i destí de la relació, el rol en cada una de les direccions i la cardinalitat que utilitzem a WebRatio (*Direct max card.*).

- Response – Thread
  - FromThread (one)
  - HasResponses (many)

- Thread – Content Type
  - HasType (one)
  - HasThreads (many)
- Thread – Specialization
  - HasSpecialization (one)
  - HasThreads (many)
- Response – User
  - FromUser (one)
  - Responses (many)
- User – Specialization
  - HasSpecialization (one)
  - HasUsers (many)
- User – Thread
  - HasThreadsPost (many)
  - PostedBy (one)
- User – Thread
  - FavoritedBy (many)
  - HasFavorites (many)

Per a tal de crear el *mapping* directament de les classes a la base de dades s'utilitza la **sincronització**.

Es pot importar la info de la database per a que s'incorpori (*Reverse Engineering*) o exportar la del model (*Forward Engineering*). En aquest cas volem exportar la del model.

WebRatio genera el codi XML per tal de que l'aplicació, un cop creada en Java, pugui utilitzar Hibernate per a la persistència de les dades. Podem trobar el codi d'aquest mapping a /WEB-INF/classes/com/webratio/webapp del projecte en qüestió o a través del menú contextual de l'entitat amb la instrucció *Open generated code*.

7.1.3 Inicialització del contingut

En el cas del projecte és necessari inicialitzar els possibles valors dels tipus de contingut i de les especialitats per a que funcioni correctament.

Aquesta inicialització s'ha de fer manualment a la base de dades o des de WebRatio hi ha l'opció d'executar arxius .sql sobre la base de dades que decidim amb la instrucció *Execute SQL*.

7.2 Transaccions

Dins de WebRatio, la transacció tan sols indica que un conjunt d'operacions s'han de fer totes juntes. Per tal de crear aquest efecte, s'utilitza un contenidor de tipus *Operation Group* indicant que volem que aquest grup d'operacions s'executi de forma transaccional.

Dins d'aquest marc podem introduir les operacions que volem executar però no aconseguim el model transaccional que tenim descrit al UML.

En primera instància, es va provar de traduir les transaccions amb *Operation Groups* però això no permetia la reutilització de transaccions si volíem executar-les des de diferents llocs.

Imatge clarificadora del que estic comentant.

En el nostre model de transaccions, el que volem és tenir un mòdul on introduir uns paràmetres d'entrada, executar-lo i obtenir una sortida. Això ho podem fer creant mòduls (*Modules*) dins de WebRatio. Aquests ens permetran utilitzar unes noves unitats anomenades *Collectors* que ens serviran de punt d'entrada i de sortida de la nostra transacció podent incorporar dades. D'aquesta manera, en un *Input Collector* podem agrupar totes les dades necessàries per a inicialitzar la transacció o paràmetres d'entrada i en els *OK/KO Collector* podem posar dades en cas d'èxit per a obtenir-los com a resultat o d'error per a tractar des de fora.

Per a crear els mòduls, primer es necessita una *Module View* que ens permetrà crear a dins els diferents models. Aquesta vista la podem crear des de la pestanya *Project* i l'anomenarem Forum Transactions.

Aquest sistema està millor explicat en l'apartat d'extracció de patrons ja que és ens servirà per a totes les transaccions de l'UML.

A continuació s'explica el resultat de la traducció de cada una de les transaccions d'UML a WebRatio.

7.2.1 TxAddFavorite

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Add Favorite.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

- oidThread
- oidUser

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

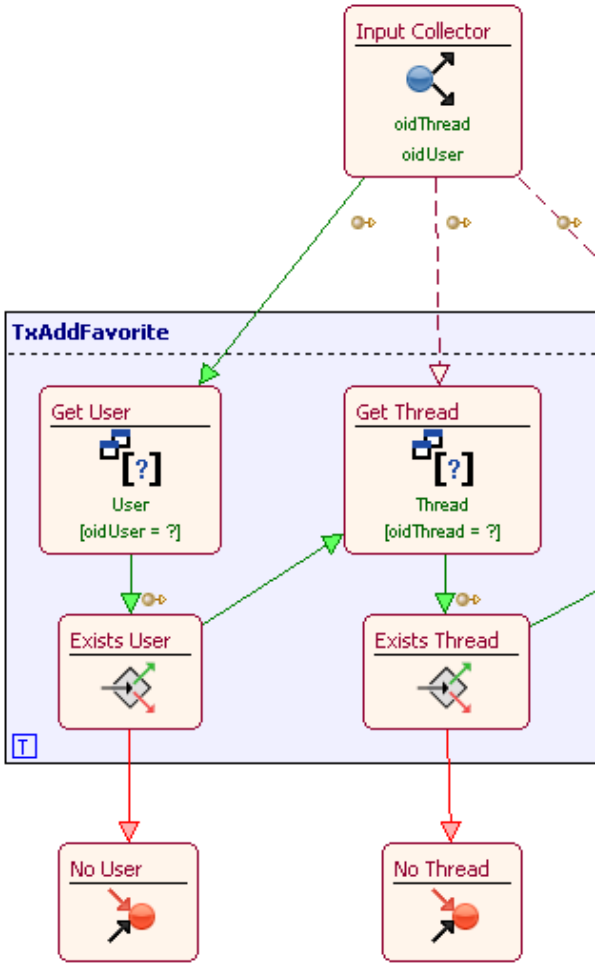
El primer que fem és comprovar els paràmetres. En aquest cas hem de tenir en compte que l'usuari i el tema de discussió siguin vàlids.

A l'UML, hauríem d'agafar les instàncies de l'usuari i del tema de discussió perquè les necessitem a la creació de la relació (unint l'usuari a la llista d'usuaris que han fet favorit el tema del tema i al revés). En el cas de WebRatio només necessitem l'identificador de les dues instàncies per a fer la relació directament ja que és una relació N-N i a nivell de base de dades estem omplint una taula diferent. Tot i així, en els dos casos accedirem a la base de dades per a extreure la instància, i com en cas de no existir retorna *null*, haurem de tractar el resultat.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari de la base de dades.
    - Afegim una *Key Condition* a l'entitat User.
    - Unim l'*Input Collector* amb Get User amb un OK Link.
        - Coupling:
            - oidUser amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem No User amb un KO Link.
- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure l'usuari de la base de dades.
    - Afegim una *Key Condition* a l'entitat Thread.
    - Unim Exists User amb Get Thread amb un OK Link.
    - Unim l'*Input Collector* amb Get Thread amb un Transport Link.
        - Coupling:
            - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
    - Unim Get Thread amb Exists Thread amb un OK Link.
        - Coupling:

- oidThread amb l'*Input*.

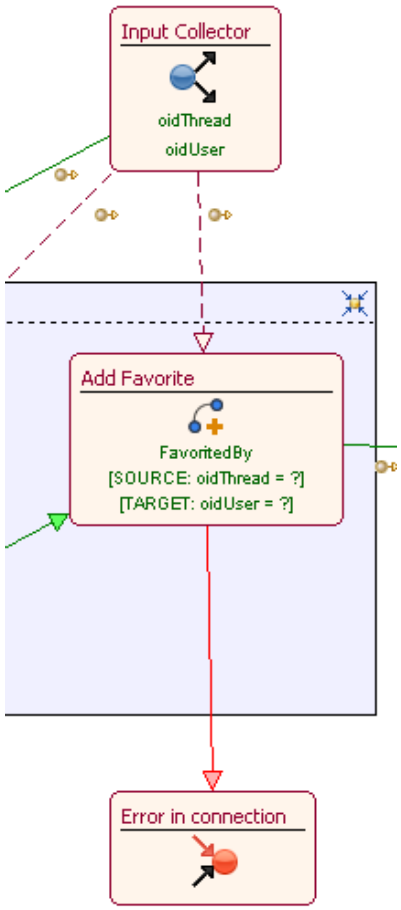
- Unim Exists Thread amb un *KO Collector* que anomenarem No Thread amb un KO Link.



Il·lustració 14: TxAddFavorite. Comprovació d'usuari i tema de discussió (WebML)

Un cop sabem que els dos identificadors són vàlids utilitzarem una *Connect Unit* per a crear la relació entre les dues instàncies. La *Connect Unit* s'utilitza en les relacions N-N del model Entity-Relationship en el que basem l'aplicació. Donada una relació i els dos identificadors (origen i destí) l'afegeix.

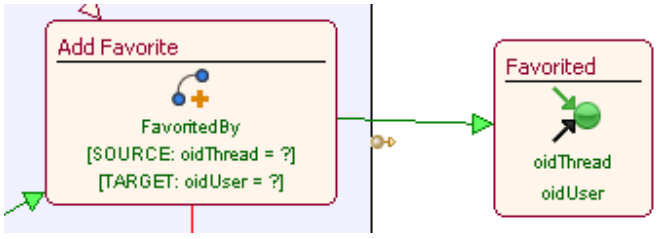
- Creem una *Connect Unit* (Add Favorite) de la relació FavoritedBy entre Thread i User.
  - Unim Exists Thread amb Add Favorite amb un OK Link.
  - Unim l'*Input Collector* amb Add Favorite amb un Transport Link.
    - Coupling:
      - oidThread amb KeyCondition[oidThread][Thread][Source].
      - oidUser amb KeyCondition[oidUser][User][Target].
  - Unim Add Favorite amb un *OK Collector* que anomenarem Favorited amb un OK Link.
  - Unim Exists Thread amb un *KO Collector* que anomenarem Error in connection amb un KO Link.



Il·lustració 15: TxAddFavorite. Creació de la relació (WebML)

Per a acabar, un cop relacionades les instàncies, modifiquem el *OK Collector* per a poder tenir els paràmetres de sortida de la transacció.

- Afegim un *Output Collector Parameter* a Favorited amb el nom oidUser que és un paràmetre de sortida de la transacció.
- Afegim un *Output Collector Parameter* a Favorited amb el nom oidThread que és un paràmetre de sortida de la transacció.
    - Unim Add Favorite amb Favorited amb un OK Link.
        - Coupling:
            - Source oidThread [Thread] amb oidThread.
            - Target oidUser [User] amb oidUser.



Il·lustració 16: TxAddFavorite. OK Collector. (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.2 TxCreateResponse

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Publish Response.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

- content
- oidUser
- oidThread

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

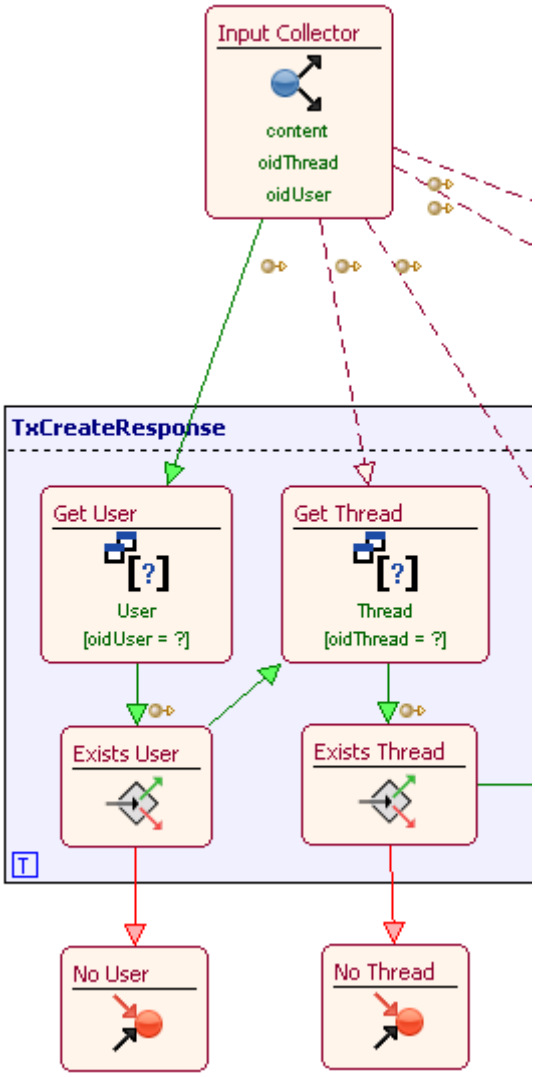
El primer que fem és comprovar els paràmetres. En aquest cas hem de tenir en compte que l'usuari i el tema de discussió siguin vàlids a més de que el contingut no sigui buit.

A l'UML, hauríem d'agafar les instàncies de l'usuari i del tema de discussió perquè les necessitem a la creació de la resposta. En el cas de WebRatio només necessitem l'identificador de la instància i no la instància en si mateixa per a poder crear-la. Tot i així, en els dos casos accedirem a la base de dades per a extreure la instància, però com en cas de no existir retorna *null*, haurem de tractar el resultat.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari que publica la resposta.
    - Afegim una *Key Condition* a l'entitat User.
    - Unim l'*Input Collector* amb Get User amb un OK Link.
        - Coupling:
            - oidUser amb la *Key Condition.*
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem No User amb un KO Link.
- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure el tema on es vol publicar la resposta.
    - Afegim una *Key* Condition a l'entitat User.
    - Unim Exists User amb Get Thread amb un OK Link.
    - Unim l'*Input Collector* amb Get Thread amb un Transport Link.
        - Coupling:
            - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
    - Unim Get Thread amb Exists Thread amb un OK Link.
        - Coupling:

- oidThread amb l'*Input*.

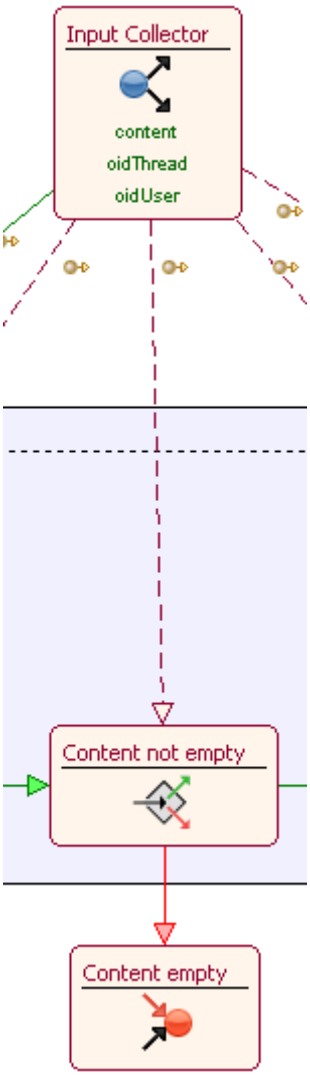
- Unim Exists Thread amb un *KO Collector* que anomenarem No Thread amb un KO Link.



Il·lustració 17: TxCreateResponse. Comprovació d'usuari i tema de discussió (WebML)

Per a comprovar que el contingut no sigui buit utilitzarem una *Is Not Null Unit*.

- Creem una *Is Not Null Unit* (Content not empty). Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
  - Unim Exists Thread amb Content not empty amb un OK Link.
  - Unim l'*Input Collector* amb Content not empty amb un Transport Link.
    - Coupling:
      - content amb l'*Input*.
  - Unim Content not empty amb un *KO Collector* que anomenarem Content empty amb un KO Link.



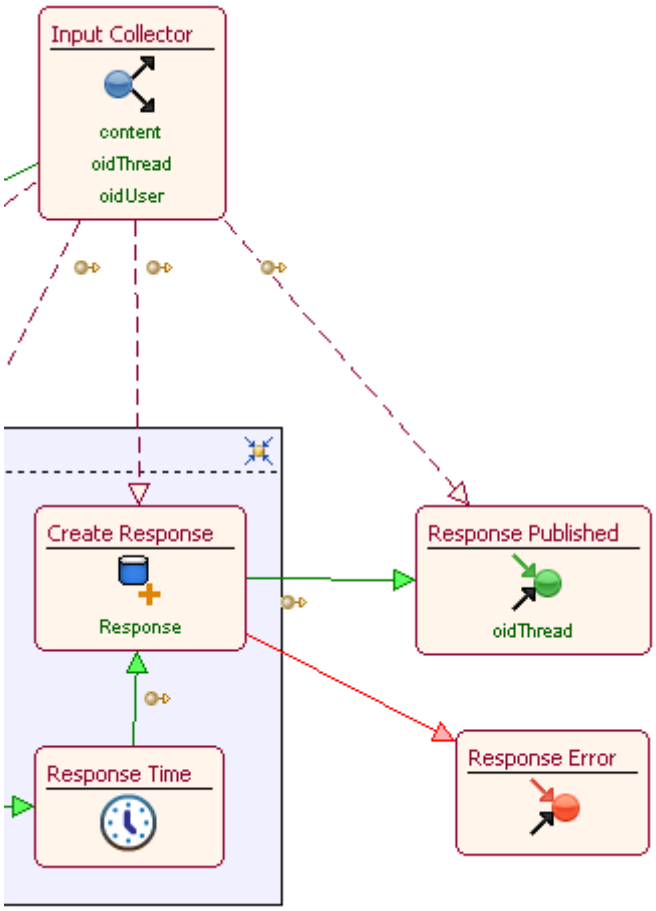
Il·lustració 18: TxCreateResponse. Comprovació del contingut (WebML)

Igual que amb la creació del tema de discussió, hem d'agafar la data abans de la creació ja que la unitat de creació ha de tenir tots els paràmetres necessaris i no ho podem fer dins la creadora com a l'UML.

- Creem una *Time Unit* (Response Time) per a accedir al temps actual del sistema.
  - Unim Content not empty amb Response Time amb un OK Link.
- Creem una *Create Unit* (Create Response) de l'entitat Response.
  - Unim Response Time amb Create Response amb un OK Link.
    - Coupling:
      - Current/Input Timestamp amb date.
  - Unim l'*Input Collector* amb Create Response amb un Transport Link.
    - Coupling:
      - content amb content.
      - oidThread amb Thread.oidThread(FromThread).
      - oidUser amb User.oidUser(FromUser).

- Unim Create Response amb un *OK Collector* que anomenarem Response Published amb un OK Link.
- Unim Create Response amb un *KO Collector* que anomenarem Response Error amb un KO Link.

- Afegim un *Output Collector Parameter* a Response Published amb el nom oidThread que és el paràmetre de sortida de la transacció.
  - Unim l'*Input Collector* amb Response Published amb un Transport Link.
    - Coupling:
      - oidThread amb oidThread.



Il·lustració 19: TxCreateResponse. Creació de resposta i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.3 TxCreateThread

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Publish Thread.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

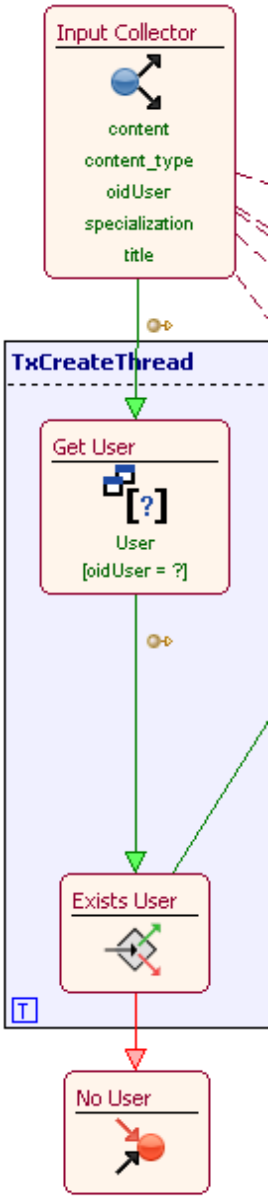
- content
- content_type
- oidUser
- specialization
- title

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

El primer que fem és comprovar els paràmetres. En aquest cas hem de tenir en compte que l'usuari, l'especialitat i el tipus de contingut siguin vàlids.

A l'UML, en els tres casos necessitem agafar les instàncies perquè les necessitem a la creació del tema de discussió. En el cas de WebRatio només necessitem l'identificador de la instància i no la instància en si mateixa per a poder crear-lo. Tot i així, en els tres casos accedirem a la base de dades per a extreure la instància, però com en cas de no existir retorna *null*, haurem de tractar el resultat.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari que crea el fil de discussió.
    - Afegim una *Key Condition* a l'entitat User.
    - Unim l'*Input Collector* amb Get User amb un OK Link.
        - Coupling:
            - oidUser amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem No User amb un KO Link.

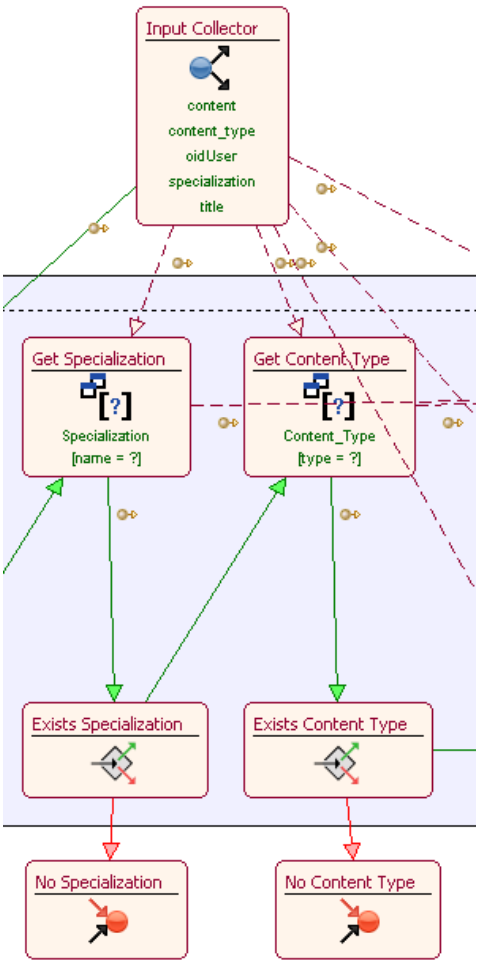


Il·lustració 20: TxCreateThread. Comprovació d'usuari (WebML)

A partir d'aquí fem el mateix amb la validació de l'especialitat i del tipus de contingut.

- Creem una *Selector Unit* (Get Specialization) de l'entitat Specialization per a extreure l'especialitat pel nom d'entrada.
    - Afegim una *Attribute Condition* per a l'atribut "name" de l'entitat Specialization.
    - Unim Exists User amb la unitat Get Specialization amb un OK Link.
    - Unim l'*Input Collector* amb Get Specialization amb un Transport Link.
        - Coupling:
            - specialization amb l'*Attribute Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Specialization).
    - Unim Get Specialization amb Exists Specialization amb un OK Link.

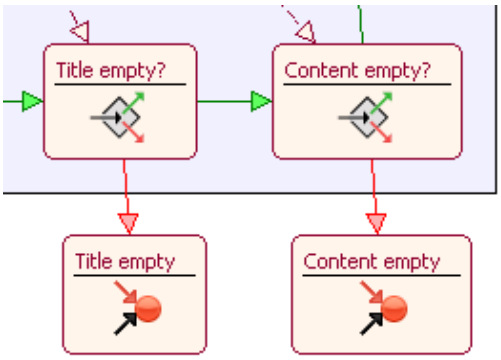
        - Coupling:
            - oidSpecialization amb l'*Input*.
    - Unim Exists Specialization amb un *KO Collector* que anomenarem No Specialization amb un KO Link.
- Creem un *Selector Unit* (Get Content Type) de l'entitat Content_Type per a extreure el tipus de contingut pel nom d'entrada.
    - Afegim una *Attribute Condition* per a l'atribut "type" de l'entitat Content_Type.
    - Unim Exists Specialization amb Get Content Type amb un OK Link.
    - Unim l'*Input Collector* amb Get Content Type amb un Transport Link.
        - Coupling:
            - content_type amb l'*Attribute Condition.*
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Content Type).
    - Unim Get Content Type amb Exists Content Type amb un OK Link.
        - Coupling:
            - oidContentType amb l'*Input*.
    - Unim Exists Content Type amb un *KO Collector* que anomenarem No Content Type amb un KO Link.



Il·lustració 21: TxCreateThread. Comprovació d'especialització i tipus de contingut (WebML)

Com a últimes comprovacions, mirem que ni el títol ni el contingut siguin *null*.

- Creem una *Is Not Null Unit* (Title empty?) per a veure si el paràmetre del títol és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Title empty? amb un Transport Link.
        - Coupling:
            - title amb l'*Input*.
    - Unim Exists Content Type amb Content empty? amb un OK Link.
    - Unim Title empty? amb un *KO Collector* que anomenarem Title Empty amb un KO Link.
- Creem una *Is Not Null Unit* (Content empty?) per a veure si el paràmetre del contingut és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Content empty? amb un Transport Link.
        - Coupling:
            - content amb l'*Input*.
    - Unim Title empty? amb Content empty? amb un OK Link.
    - Unim Content empty? amb un *KO Collector* que anomenarem Content Empty amb un KO Link.

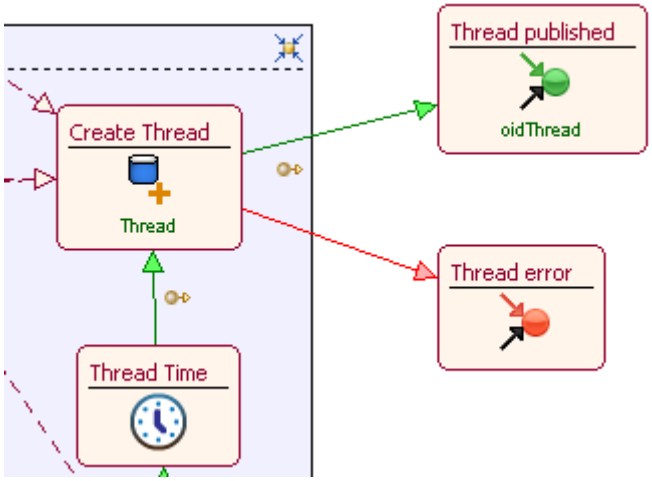


Il·lustració 22: TxCreateThread. Comprovació de títol i contingut (WebML)

Un cop fetes les comprovacions i tenint ja els diferents identificadors de les instàncies necessàries agafarem el temps actual i crearem la instància del tema de discussió.

- Creem una *Time Unit* (Thread Time) per a accedir al temps actual del sistema.
    - Unim Content empty? amb Thread Time amb un OK Link.
- Creem una *Create Unit* (Create Thread) de l'entitat Thread per a crear la instància del tema de discussió.
    - Unim Thread Time amb Create Thread amb un OK Link.
        - Coupling:
            - Current/Input Timestamp amb date.
    - Unim l'*Input Collector* amb Create Thread amb un Transport Link.
        - Coupling:
            - content amb content.
            - title amb title.

        - oidUser amb User.oidUser(PostedBy).
    - Unim Get Specialization amb Create Thread amb un Transport Link.
        - Coupling:
            - oidSpecialization amb Specialization.oidSpecializaton(HasSpecialization).
    - Unim Get Content Type amb Create Thread amb un Transport Link.
        - Coupling:
            - oidContentType amb ContentType.oidContentType(HasType).
    - Unim Create Thread amb un *KO Collector* que anomenarem Thread Error amb un KO Link.
    - Unim Create Thread amb un *OK Collector* que anomenarem Thread Published amb un OK Link.
- Afegim un *Output Collector Parameter* a Thread Published amb el nom oidThread que és el paràmetre de sortida de la transacció.
    - Modifiquem el OK Link entre Create Thread i Thread Published.
        - Coupling:
            - oidThread amb oidThread.



Il·lustració 23: TxCreateThread. Creació del tema de discussió i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.4 TxCreateUser

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Register.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

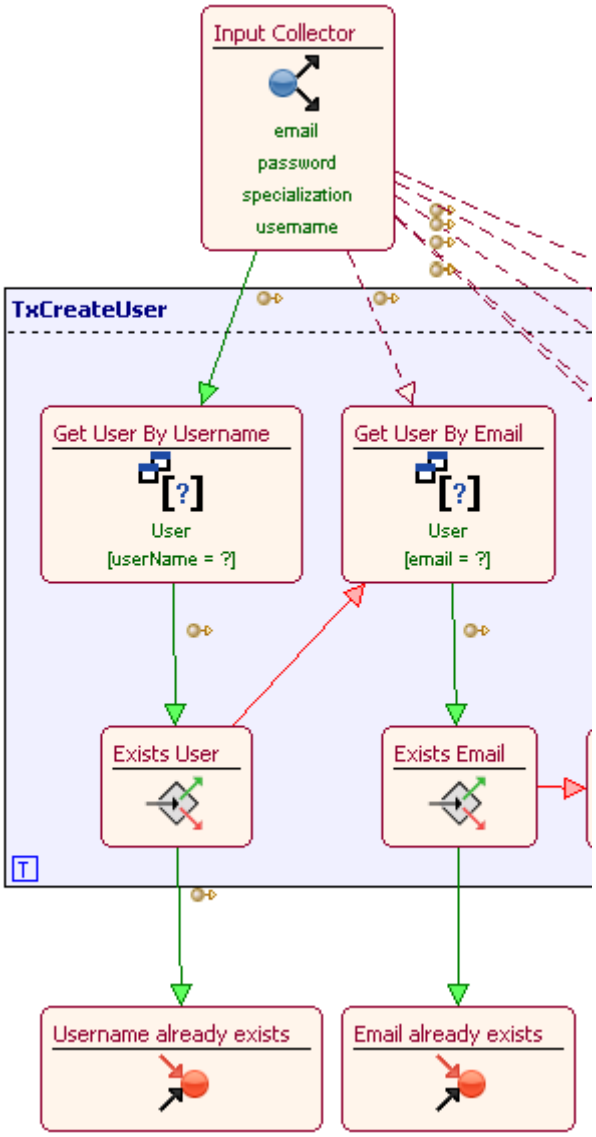
- email
- password
- specialziation
- username

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

El primer de tot és assegurar-nos de que tant l'username com l'email i el password no són buits. No podem tenir un usuari sense nom, email o password.

A continuació comprovem que no existeixi ja un usuari amb el mateix nom o el mateix email.

- Creem una *Selector Unit* (Get User By Username) de l'entitat User per a extreure l'usuari de la base de dades.
    - Afegim una *Attribute Condition* per a l'atribut "userName" de l'entitat User.
    - Unim l'*Input Collector* amb Get User By Username amb un OK Link.
        - Coupling:
            - userName amb l'*Attribute Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User By Username amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem Username already exists amb un KO Link.
- Creem una *Selector Unit* (Get User By Email) de l'entitat User per a extreure l'usuari de la base de dades.
    - Afegim una *Attribute Condition* per a l'atribut "email" de l'entitat User.
    - Unim Get User By Username amb Get User By Email amb un OK Link.
        - Coupling:
            - email amb l'*Attribute Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Email).
    - Unim Get User By Email amb Exists Email amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem Email already exists amb un KO Link.



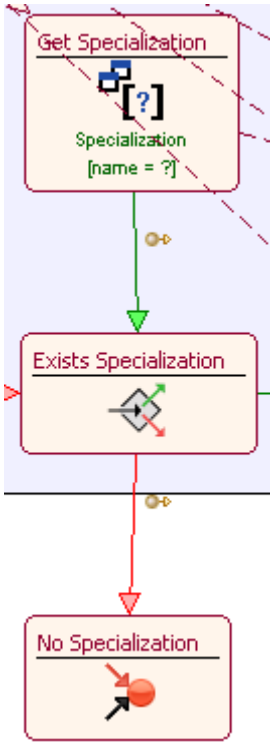
Il·lustració 24: TxCreateUser. Comprovació d'usuari i e-mail (WebML)

Ens hem de fixar que aquesta vegada volem el comportament contrari a la resta d'operacions quan usem la *Is Not Null Unit*, que és fer alguna cosa quan es *null*.

Un cop comprovat que el nom d'usuari i l'e-mail són correctes, comprovem que l'especialitat també ho és. A l'UML, hauríem d'agafar la instància sencera per a fer la relació però a WebRatio només necessitem l'identificador de la instància per a poder-ho fer. Com en cas de no existir retorna *null*, haurem de tractar el resultat.

- Creem una *Selector Unit* (Get Specialization) de l'entitat Specialization per a extreure l'especialitat pel nom d'entrada.
    - Afegim una *Attribute Condition* per a l'atribut "name" de l'entitat Specialization.
    - Unim Exists Email amb la unitat Get Specialization amb un KO Link.
    - Unim l'*Input Collector* amb Get Specialization amb un Transport Link.
        - Coupling:

        - specialization amb l'*Attribute Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Specialization).
    - Unim Get Specialization amb Exists Specialization amb un OK Link.
        - Coupling:
            - oidSpecialization amb l'*Input*.
    - Unim Exists Specialization amb un *KO Collector* que anomenarem No Specialization amb un KO Link.

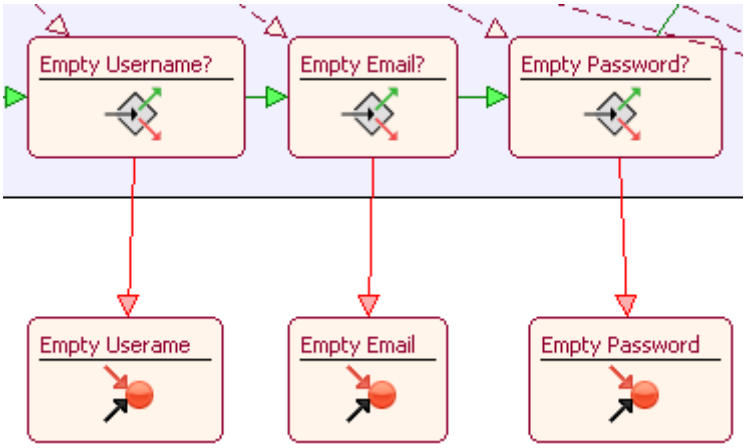


Il·lustració 25: TxCreateUser. Comprovació d'especialitat (WebML)

Haurem de comprovar que tots els paràmetres amb els que s'ha cridat la transacció no són *null* abans de crear la nova instància.

- Creem una *Is Not Null Unit* (Empty Username?) per a veure si el paràmetre del nom d'usuari és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Empty Username? amb un Transport Link.
        - Coupling:
            - username amb l'*Input*.
    - Unim Exists Specialization amb Empty Username? amb un OK Link.
    - Unim Empty Username? amb un *KO Collector* que anomenarem Empty Username amb un KO Link.
- Creem una *Is Not Null Unit* (Empty Email?) per a veure si el paràmetre del e-mail és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Empty Email? amb un Transport Link.

        - Coupling:
            - email amb l'*Input*.
    - Unim Empty Username? amb Empty Email? amb un OK Link.
    - Unim Empty Email? amb un *KO Collector* que anomenarem Empty Email amb un KO Link.
- Creem una *Is Not Null Unit* (Empty Password?) per a veure si el paràmetre de la contrasenya és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Empty Password? amb un Transport Link.
        - Coupling:
            - password amb l'*Input*.
    - Unim Empty Username? amb Empty Password? amb un OK Link.
    - Unim Empty Password? amb un *KO Collector* que anomenarem Empty Password amb un KO Link.



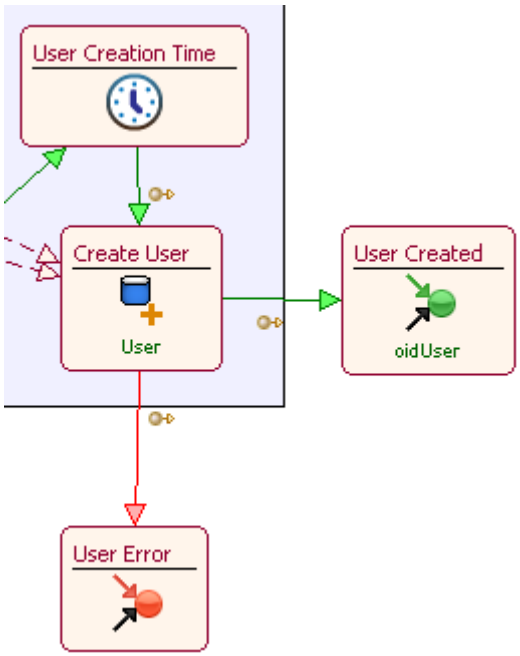
Il·lustració 26: TxCreateUser. Comprovació de paràmetres buits (WebML)

Per a completar la creació de la instància d'usuari, necessitem l'hora actual del sistema que ens serveix per a data de creació de l'usuari i últim login ja que entrarà automàticament al sistema.

- Creem una *Time Unit* (User Creation Time) per a accedir al temps actual del sistema.
    - Unim Empty Password? amb User Creation Time amb un OK Link.
- Creem una *Create Unit* (Create User) de l'entitat User per a crear la instància de l'usuari.
    - Unim User Creation Time amb Create User amb un OK Link.
        - Coupling:
            - Current/Input Timestamp amb creation_time.
            - Current/Input Timestamp amb last_connection.
    - Unim l'*Input Collector* amb Create User amb un Transport Link.
        - Coupling:
            - email amb email.
            - password amb password.
            - username amb userName.

- Unim Get Specialization amb Create User amb un Transport Link.
  - Coupling:
    - oidSpecialization amb Specialization.oidSpecializaton(HasSpecialization).
- Unim Create User amb un *KO Collector* que anomenarem User Error amb un KO Link.
- Unim Create User amb un *OK Collector* que anomenarem User Created amb un OK Link.

- Afegim un *Output Collector Parameter* a User Created amb el nom oidUser que és el paràmetre de sortida de la transacció.
  - Modifiquem el OK Link entre Create User i User Created.
    - Coupling:
      - oidUser amb oidUser.



Il·lustració 27: TxCreateUser. Creació de l'usuari amb la data i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.5 TxGetContentTypes

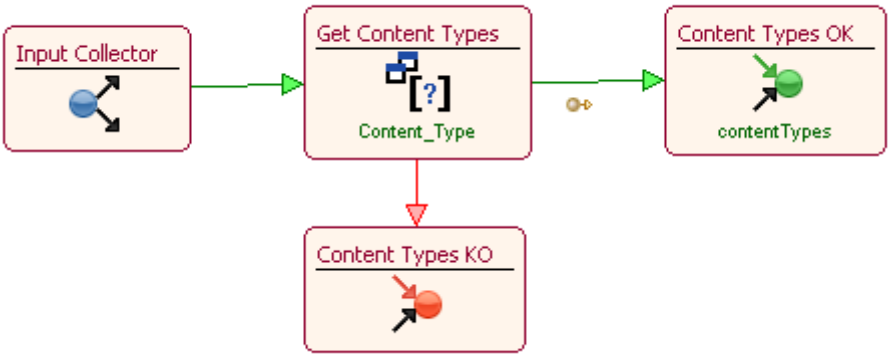
Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Get Content Types.

Dins del mòdul crearem un *Input Collector* sense cap paràmetre que serà el punt d'inici del mòdul.

En aquest cas que nomes estem extraient informació de la base de dades i que no farem cap modificació, no és necessari englobar les operacions en una transacció. De fet, només tenim una operació de consulta a fer.

No utilitzarem cap selector a l'hora d'extreure tots els tipus de contingut de la base de dades. D'aquesta forma podem extreure tots els valors existents.

- Creem una *Selector Unit* (Get Content Types) de l'entitat Content_Type.
    - Unim Get Content Types amb un *KO Collector* que anomenarem Content Types OK amb un KO Link.
    - Unim Get Content Types amb un *OK Collector* que anomenarem Content Types KO amb un OK Link.
- Afegim un *Output Collector Parameter* a Content Types OK amb el nom contentTypes que és el paràmetre de sortida de la transacció.
    - Modifiquem el OK Link entre Get Content Types i Content Types OK.
        - Coupling:
            - types amb contentTypes.



Il·lustració 28: TxGetContentTypes. Diagrama complet (WebML)

7.2.6 TxGetSpecializations

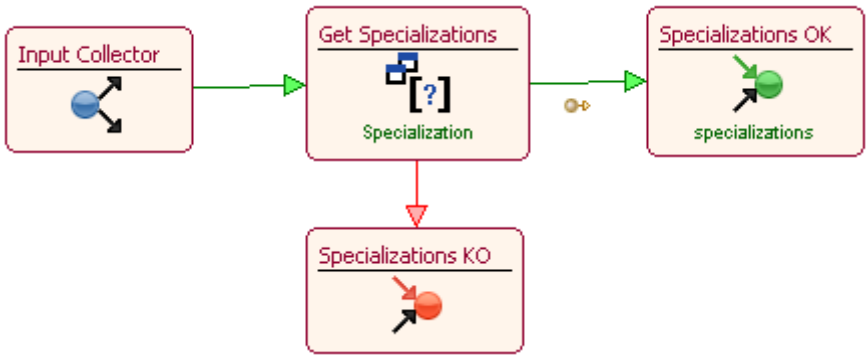
Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Get Specializations.

Dins del mòdul crearem un *Input Collector* sense cap paràmetre que serà el punt d'inici del mòdul.

En aquest cas que nomes estem extraient informació de la base de dades i que no farem cap modificació, no és necessari englobar les operacions en una transacció. De fet, només tenim una operació de consulta a fer.

No utilitzarem cap selector a l'hora d'extreure tots els tipus de contingut de la base de dades. D'aquesta forma podem extreure tots els valors existents.

- Creem una *Selector Unit* (Get Specializations) de l'entitat Specialization.
    - Unim Get Specializations amb un *KO Collector* que anomenarem Specializations OK amb un KO Link.
    - Unim Get Specializations amb un *OK Collector* que anomenarem Specializations KO amb un OK Link.
- Afegim un *Output Collector Parameter* a Specializations OK amb el nom specializations que és el paràmetre de sortida de la transacció.
    - Modifiquem el OK Link entre Get Specializations i Specializations OK.
        - Coupling:
            - name amb specializations.



Il·lustració 29: TxGetSpecializations. Diagrama complet (WebML)

7.2.7 TxModifyResponse

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Modify Response.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

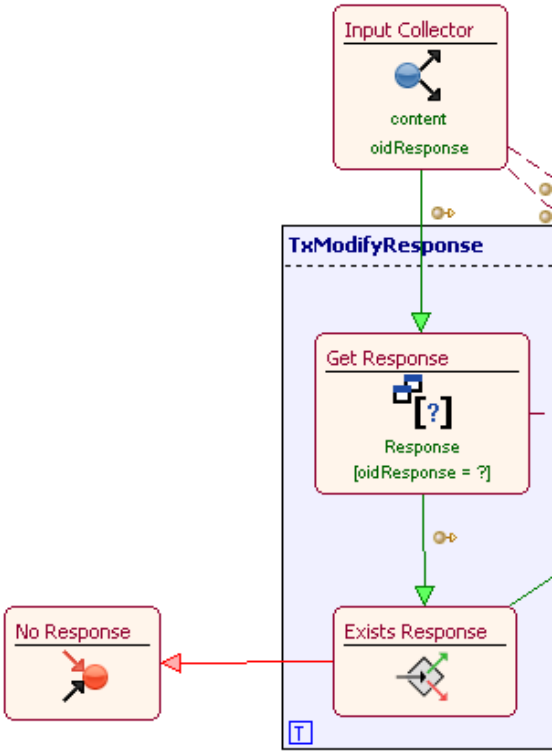
- content
- oidResponse

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

En el cas d'aquesta transacció, només es modificarà la instància en el cas de que el contingut no sigui buit. En cas contrari no es farà res durant la operació tal i com es pot veure en el diagrama UML tot i que no es tractarà com un error.

El primer és agafar la instància del tema a modificar de la base de dades. En el cas de que no existeixi, s'atura l'execució per una excepció. Un cop l'agafem, podrem veure si l'identificador és vàlid o no i a més a més sabrem de quin tema és la resposta ja que ho necessitem perquè es la variable de retorn de la transacció.

- Creem una *Selector Unit* (Get Response) de l'entitat Response per a extreure la resposta de la base de dades.
    - Afegim una *Key Condition* a l'entitat Response.
    - Unim l'*Input Collector* amb la unitat Get Response amb un OK Link.
        - Coupling:
            - oidResponse amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Response).
    - Unim Get Response amb Exists Response amb un OK Link.
        - Coupling:
            - oidResponse amb l'*Input*.
    - Unim Exists Response amb un *KO Collector* que anomenarem No Response amb un KO Link.



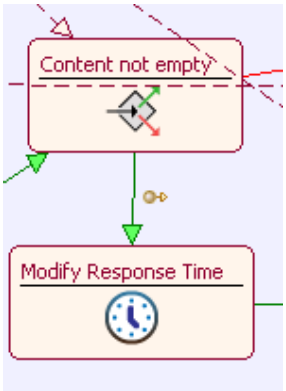
Il·lustració 30: TxModifyResponse. Comprovació de la resposta (WebML)

A continuació mirem que el paràmetre de contingut no sigui buit per tal de modificar-lo.

- Creem una *Is Not Null Unit* (Content not empty) per a veure si el paràmetre del contingut és buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Content not empty amb un Transport Link.
        - Coupling:
            - content amb l'*Input*.
    - Unim Exists Response amb Content not empty amb un OK Link.

En cas de no ser buit, volem fer la modificació a la instància.

- Creem una *Time Unit* (Modify Response Time) per a accedir al temps actual del sistema.
    - Unim Content not empty amb Modify Response Time amb un OK Link.
- Creem una *Modify Unit* (Modify Response) de l'entitat Response per a poder fer la modificació del contingut i de la data de modificació sobre la resposta.
    - Unim l'*Input Collector* amb Modify Response amb un Transport Link.
        - Coupling:
            - content amb content.
            - oidResponse amb KeyCondition[oidResponse].
    - Unim Modify Response Time amb Modify Response amb un OK Link.
        - Coupling:
            - Current/Input Timestamp amb modified_date.



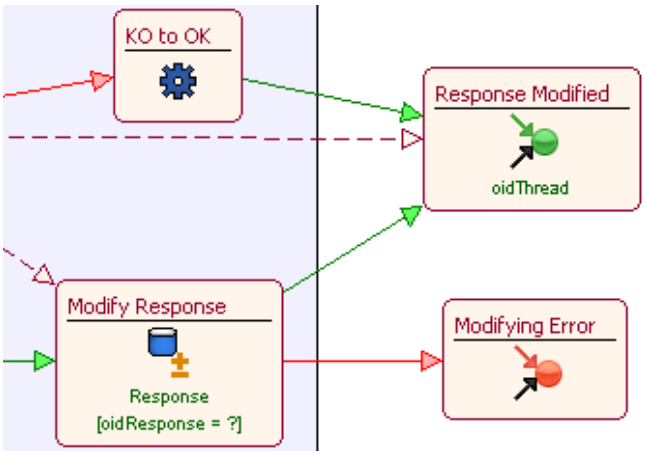
Il·lustració 31: TxModifyResponse. Comprovació de contingut i accés a la data (WebML)

Arribats a aquest punt nomes ens fa falta guardar el valor de retorn del tema de discussió al que pertany la resposta. Per a això crearem un *OK Collector* amb l'identificador com a variable i li assignarem valor tan si hem fet la modificació com si no.

- Creem un *OK Collector* que anomenarem Response Modified.
    - Unim Modify Response amb Response Modified amb un OK Link.

Com no podem unir un KO Link a un *OK Collector* (de Content not empty a Response Modified) usarem un a *No Op Operation Unit* per a fer el canvi de link. Així doncs, arribarem fins a la nova unitat amb un KO Link i farem sortir un OK Link des d'aquesta.

- Creem una *No Op Operation Unit* (KO to OK).
    - Unim Content not empty amb KO to OK amb un KO Link.
    - Unim KO to OK amb Response Modified amb un OK Link.
- Afegim un *Output Collector Parameter* a Response Modified amb el nom oidThread que és el paràmetre de sortida de la transacció.
    - Unim Get Response amb Response Modified amb un Transport Link.
        - Coupling:
            - FromThread.oidThread amb oidThread.



Il·lustració 32: TxModifyResponse. Modificació de la resposta i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.8 TxModifyThread

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Modify Thread.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

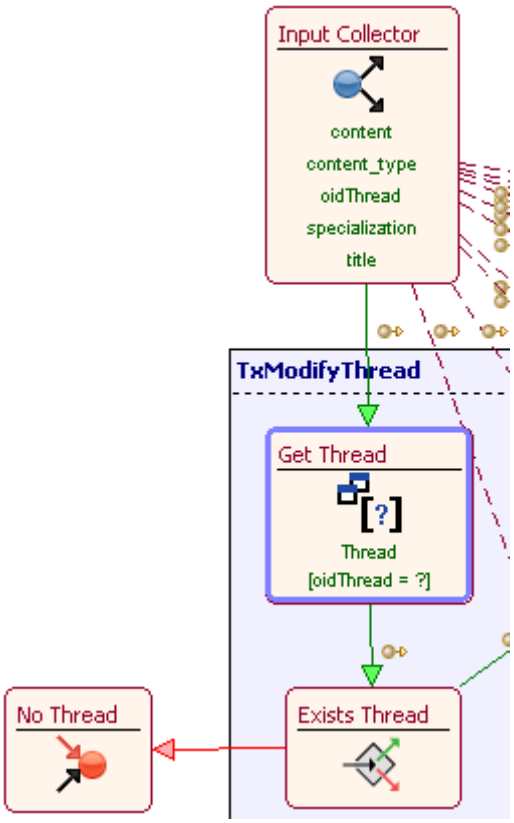
- content
- content_type
- oidThread
- specialization
- title

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

En el cas d'aquesta operació, a la transacció només modifiquem aquells paràmetres que no siguin buits. Tal i com pensem amb la persistència directe, a cada *set* que fem d'un atribut de la instància estem modificant-la i guardant aquest canvi. Per a poder recrear aquest comportament en WebRatio haurem de fer una modificació per a cada atribut després de comprovar si el paràmetre és buit o no.

El primer és agafar la instància del tema a modificar de la base de dades. En el cas de que no existeixi, s'atura l'execució per una excepció. A WebRatio només ens interessarà saber si l'identificador és vàlid o no, no necessitem més informació sobre la instància.

- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure el tema de discussió de la base de dades.
    - Afegim una *Key Condition* a l'entitat Thread.
    - Unim l'*Input Collector* amb Get Thread amb un OK Link.
        - Coupling:
            - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
    - Unim Get Thread amb Exists Thread amb un OK Link.
        - Coupling:
            - oidThread amb l'*Input*.
    - Unim Exists Thread amb un *KO Collector* que anomenarem No Thread amb un KO Link.



Il·lustració 33: TxModifyThread. Comprovació del tema de discussió (WebML)

Un cop sabem que l'identificador és vàlid, seguirem mirant un per un els paràmetres de la transacció mirant si no són *null* i és necessari que es modifiquin. L'ordre serà el mateix amb el que fem les comprovacions al UML: especialitat, tipus de contingut, títol i contingut.

En el cas de l'especialitat i del tipus de contingut, a l'UML extraiem la instància de la base de dades per a després poder fer el canvi. En aquest cas, farem el mateix però per a obtenir l'identificador, tal com s'ha explicat en la transacció TxCreateThread, ja que és l'únic que necessitem per a canviar l'assignació.

- Creem una *Is Not Null Unit* (Need to modify Specialization?) per a veure si el paràmetre de l'especialitat s'ha deixat buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim Exists Thread amb Need to modify Specialization? amb un OK Link.
    - Unim l'*Input Collector* amb Need to modify Specialization? amb un Transport Link.
        - Coupling:
            - specialization amb *Input*.

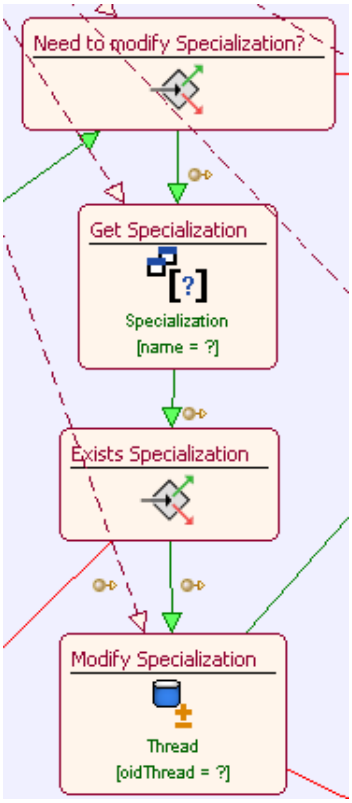
Des de Need to modify Specialization? tenim dues opcions. En el cas de ser *null* no volem fer res i en el cas de no ser-ho volem fer la modificació.

Primer farem les operacions necessàries per a fer la modificació i després enllaçarem les dues vies .

- Creem un *Selector Unit* (Get Specialization) de l'entitat Specialization per a extreure l'especialitat pel nom d'entrada.
    - Afegim una *Attribute Condition* per a l'atribut "name" de l'entitat Specialization.
    - Unim l'*Input Collector* amb Get Specialization amb un Transport Link.
        - Coupling:
            - specialization amb l'*Attribute Condition*.
    - Unim Need to modify Specialization? amb Get Specialization amb un OK Link.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Specialization).
    - Unim Get Specialization amb Exists Specialization amb un OK Link.
        - Coupling:
            - oidSpecialization amb l'*Input*.
        - Passing:
            - oidSpecialization.
    - Unim Exists Specialization amb un *KO Collector* que anomenarem No Specialization amb un KO Link.

S'ha optat per passar la informació de l'identificador de la *Selector Unit* a la *Is Not Null Unit* perquè l'utilitzarem directament a la següent operació. Es podria haver fet de amb un Transport Link de la *Selector Unit* a la *Modify Unit* i tindria el mateix efecte.

- Creem una *Modify Unit* (Modify Specialization) de l'entitat Thread per tal de poder fer la modificació de l'especialitat sobre el tema de discussió.
    - Unim Exists Specialization amb Modify Specialization amb un OK Link.
        - Coupling:
            - oidSpecialization_PASSING amb Specialization.oidSpecialization(HasSpecialization).
    - Unim l'*Input Collector* amb Modify Specialization amb un Transport Link.
        - Coupling:
            - oidThread amb KeyCondition[oidThread].
    - Unim Modify Specialization amb un *KO Collector* que anomenarem Error Modifying amb un KO Link.



Il·lustració 34: TxModifyThread. Comprovació i modificació de l'especialitat (WebML)

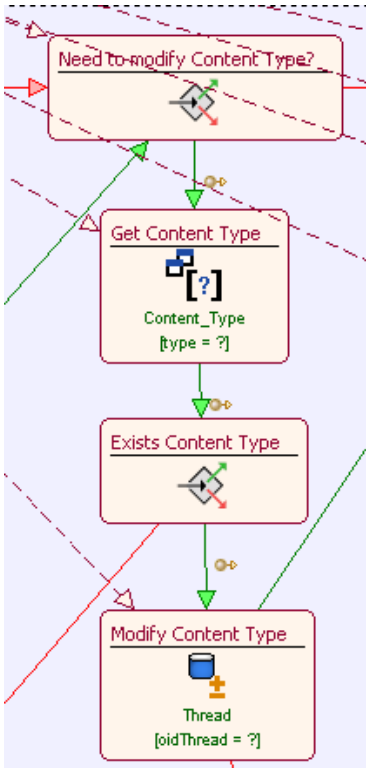
En aquest moment ja tenim tractat el canvi d'especialitat. De la mateixa manera que abans, comprovem el tipus de contingut i enllaçarem les dues vies fins a la nova unitat.

- Creem una *Is Not Null Unit* (Need to modify Content Type?) per a veure si el paràmetre del tipus de contingut s'ha deixat buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Need to modify Content Type? amb un Transport Link.
        - Coupling:
            - content_type amb *Input*.
    - Unim Need to modify Specialization amb Need to modify Content Type? amb un KO Link.
    - Unim Modify Specialization amb Need to modify Content Type? amb un OK Link.
- Creem un *Selector Unit* (Get Content Type) de l'entitat Content_Type per a extreure el tipus de contingut pel nom d'entrada.
    - Afegim una *Attribute Condition* per a l'atribut "type" de l'entitat Content_Type.
    - Unim l'*Input Collector* amb Get Content Type amb un Transport Link.
        - Coupling:
            - content_type amb l'*Attribute Condition*.
    - Unim Need to modify Content Type? amb Get Content Type amb un OK Link.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Content Type).

- Unim Get Content Type amb Exists Content Type amb un OK Link.
  - Coupling:
    - oidContentType amb l'*Input*.
  - Passing:
    - oidContentType.
- Unim Exists Content Type amb un *KO Collector* que anomenarem No Content Type amb un KO Link.

Igual que en el cas de la especialitat, s'ha optat per passar la informació de l'identificador de la *Selector Unit* a la *Is Not Null Unit* perquè l'utilitzarem directament a la següent operació. Es podria haver fet de amb un Transport Link de la *Selector Unit* a la *Modify Unit* i tindria el mateix efecte.

- Creem una *Modify Unit* (Modify Content Type) de l'entitat Thread per tal de poder fer la modificació del tipus de contingut sobre el tema de discussió.
  - Unim Exists Content Type amb Modify Content Type amb un OK Link.
    - Coupling:
      - oidContentType_PASSING amb Content_Type.oidContentType(HasType).
  - Unim l'*Input Collector* amb Modify Content Type amb un Transport Link.
    - Coupling:
      - oidThread amb KeyCondition[oidThread].
  - Unim Modify Content Type amb el *KO Collector* Error Modifying amb un KO Link.



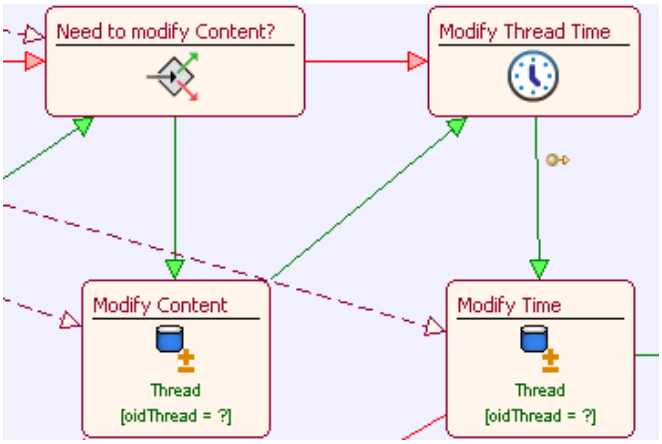
Il·lustració 35: TxModifyThread. Comprovació i modificació del tipus de contingut (WebML)

En el cas del títol i del contingut del tema de discussió, fer la modificació és molt més senzill perquè no hem de comprovar res, només si el paràmetre és buit o no i modificar-lo en funció del resultat. De la mateixa forma d'abans haurem d'enllaçar també les dues ramificacions.

- Creem una *Is Not Null Unit* (Need to modify Title?) per a veure si el paràmetre del títol s'ha deixat buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim l'*Input Collector* amb Need to modify Title? amb un Transport Link.
        - Coupling:
            - title amb *Input*.
    - Unim Modify Content Type amb Need to modify Title? amb un OK Link.
    - Unim Need to modify Content Type amb Need to modify Title? amb un KO Link.
- Creem una *Modify Unit* (Modify Title) de l'entitat Thread per tal de poder fer la modificació del títol sobre el tema de discussió.
    - Unim Need to modify Title? amb Modify Title amb un OK Link.
    - Unim l'*Input Collector* amb Modify Title amb un Transport Link.
        - Coupling:
            - oidThread amb KeyCondition[oidThread].
            - title amb title.
    - Unim Modify Title amb el *KO Collector* Error Modifying amb un KO Link.

Fem el mateix amb el contingut.

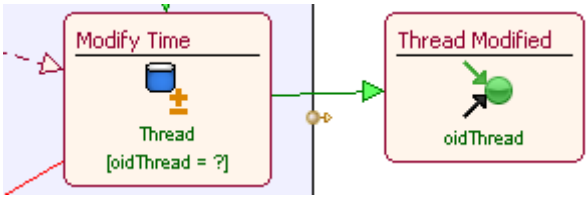
- Creem una *Is Not Null Unit* (Need to modify Content?) per a veure si el paràmetre del contingut s'ha deixat buit o no. Cal indicar que es tracti l'*Empty String* com a *null* per a que funcioni correctament.
    - Unim *l'Input Collector* amb Need to modify Content? amb un Transport Link.
        - Coupling:
            - content amb *Input*.
    - Unim Need to modify Title? amb Need to modify Content? amb un KO Link.
    - Unim Modify Title amb Need to modify Content? amb un OK Link.
- Creem una *Modify Unit* (Modify Content) de l'entitat Thread per tal de poder fer la modificació del contingut sobre el tema de discussió.
    - Unim Need to modify Content? amb Modify Content amb un OK Link.
    - Unim l'*Input Collector* amb Modify Content amb un Transport Link.
        - Coupling:
            - oidThread amb KeyCondition[oidThread].
            - content amb content.
    - Unim Modify Content amb el *KO Collector* Error Modifying amb un KO Link.



Il·lustració 36: TxModifyThread. Comprovació i modificació del contingut i accés a la data (WebML)

Ara per ara ja hem fet tots els canvis sobre el tema de discussió segons els paràmetres d'entrada. En aquest moment només fa falta canviar la data de modificació i retornar l'identificador del tema de discussió.

- Creem una *Time Unit* (Modify Thread Time) per a accedir al temps actual del sistema.
  - Unim Need to modify Content? amb Modify Thread Time amb un KO Link.
  - Unim Modify Content amb Modify Thread Time amb un OK Link.
- Creem una *Modify Unit* (Modify Time) de l'entitat Thread per tal de poder fer la modificació la data de modificació sobre el tema de discussió.
  - Unim Modify Thread Time amb Modify Time amb un OK Link.
    - Coupling:
      - Current/Input Timestamp amb modified_date.
  - Unim l'*Input Collector* amb Modify Time amb un Transport Link.
    - Coupling:
      - oidThread amb KeyCondition[oidThread].
  - Unim Modify Content amb el *KO Collector* Error Modifying amb un KO Link.
  - Unim Create Thread amb un *OK Collector* que anomenarem Thread Modified amb un OK Link.
- Afegim un *Output Collector Parameter* a Thread Modified amb el nom oidThread que és el paràmetre de sortida de la transacció.
  - Modifiquem el OK Link entre Modify Time i Thread Modified.
    - Coupling:
      - oidThread amb oidThread.



Il·lustració 37: TxModifyThread. Modificació del temps i OK Collector (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.9 TxModifyUser

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Remove Favorite.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

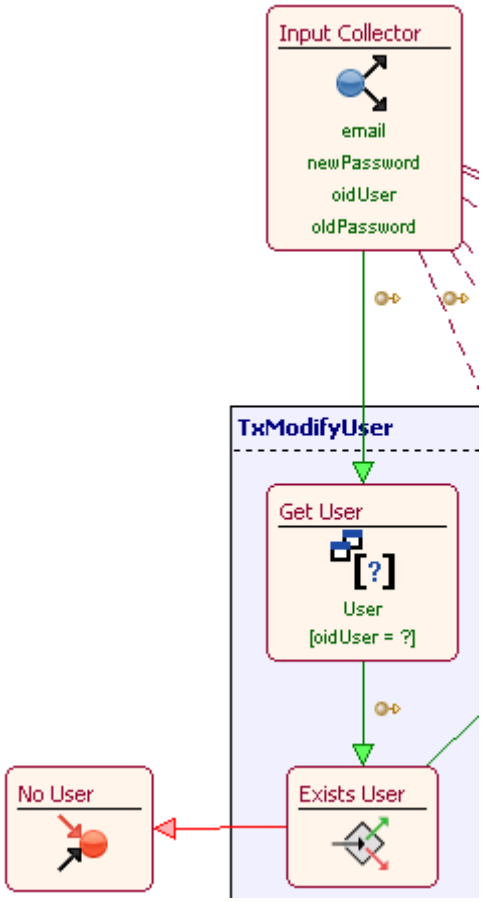
- email
- newPassword
- oidUser
- oldPassword

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

En aquesta transacció es miraran de modificar algunes dades de l'usuari (e-mail i contrasenya) en el cas de que els paràmetres siguin correctes i no estiguin buits. En el cas d'estar buits, no es farà la modificació d'aquell atribut. En el cas de la contrasenya, el nou ha de tenir un valor assignat i l'antic correspondre a l'actual contrasenya de l'usuari que volem modificar.

El primer que fem és comprovar els paràmetres. En aquest cas hem de tenir en compte que l'usuari ha de ser vàlid.

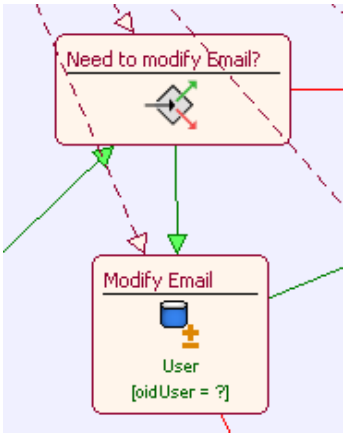
- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari de la base de dades.
    - Afegim una *Key Condition* a l'entitat User.
    - Unim l'*Input Collector* amb la unitat Get User amb un OK Link.
        - Coupling:
            - oidUser amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists User amb un *KO Collector* que anomenarem No User amb un KO Link.



Il·lustració 38: TxModifyUser. Comprovació de l'usuari (WebML)

Per saber si hem de modificar l'e-mail o la contrasenya hem de saber si el paràmetre és buit o no i modificar-lo en funció del resultat. En els casos com aquest on tenim més d'una opció possible (si és buit o no) amb operacions diferents haurem d'usar un *Parameter Collector* per a poder gestionar-ho. Començarem mirant si s'ha de modificar l'e-mail.

- Creem una *Is Not Null Unit* (Need to modify Email?) per a saber si cal o no canviar l'e-mail de l'usuari.
  - Unim Exists User amb Need to modify Email? amb un OK Link.
  - Unim l'*Input Collector* amb Need to modify Email? amb un Transport Link.
    - Coupling:
      - email amb l'*Input*.
- Creem una *Modify Unit* (Modify Email) de l'entitat User per tal de modificar l'usuari si l'e-mail no era *null*.
  - Unim Need to modify Email? amb Modify Email amb un OK Link.
  - Unim l'*Input Collector* amb Modify Email amb un Transport Link.
    - Coupling:
      - email amb email.
      - oidUser amb KeyCondition[oidUser].
  - Unim Modify Email amb un KO *Collector* que anomenarem Modifying Error amb un KO Link.



Il·lustració 39: TxModifyUser. Comprovació i modificació de l'e-mail (WebML)

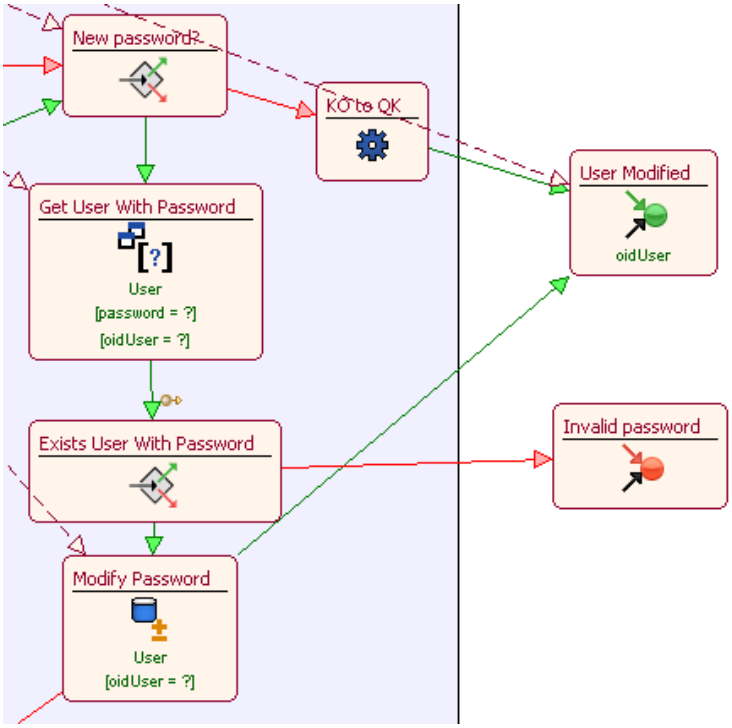
Un cop ja tenim modificat si és necessari l'e-mail de l'usuari, es segueix el mateix sistema per a fer el canvi de contrasenya però aquest cop és una mica més complicat per la lògica del canvi. El canvi només s'efectuarà en el cas de que s'hagi introduït una nova contrasenya. Haurem d'accedir a la base de dades a extreure l'usuari per veure si la contrasenya es correspon. Això ho podem fer amb una comparació de l'atribut de l'usuari o accedint a la base de dades amb més d'una condició (nom d'usuari i contrasenya) i veure si hi ha algun resultat. S'ha optat per a aquesta segona, aprofitant així la potència de l'SQL.

- Creem una *Is Not Null Unit* (New password?) per a saber si cal o no canviar la contrasenya de l'usuari.
    - Unim el Need to modify Email? amb New password? amb un KO Link.
    - Unim el Modify Email amb New password? amb un OK Link.
    - Unim l'*Input Collector* amb New password? amb un Transport Link.
        - Coupling: newPassword amb l'*Input*.
- Creem una *Selector Unit* (Get User With Password) de l'entitat User per tal de veure si la contrasenya antiga coincideix amb la de l'usuari que volem modificar.
    - Afegim una *Attribute Condition* a Get User With Password sobre l'atribut oidUser i l'anomenarem oidUser.
    - Afegim una *Attribute Condition* a Get User With Password sobre l'atribut password i l'anomenarem password.
    - Unim New password? amb Get User With Password amb un OK Link.
    - Unim l'*Input Collector* amb Get User With Password amb un Transport Link.
        - Coupling:
            - oidUser amb oidUser.
            - oldPassword amb password.
- Creem una *Is Not Null Unit* (Exists user with password) per a comprovar si la contrasenya corresponia a l'usuari.
    - Unim Get User With Password amb Exists user with password amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
    - Unim Exists user with password amb un KO *Collector* que anomenarem Invalid Password amb un KO Link.

- Creem una *Modify Unit* (Modify Password) de l'entitat User per tal de modificar la contrasenya de l'usuari.
    - Unim Exists user with password amb Modify Password amb un OK Link.
    - Unim l'*Input Collector* amb Modify Password amb un Transport Link.
        - Coupling:
            - oidUser amb KeyCondition[oidUser].
            - newPassword amb password.
    - Unim Modify Email amb Modifying Error amb un KO Link.
- Creem un *OK Collector* (User Modified) que indicarà la finalització correcte de la transacció.
    - Unim Modify Password amb User Modified amb un OK Link.

Com no podem unir un KO Link a un *OK Collector* (de New Password? a User Modified) usarem un a *No Op Operation Unit* per a fer el canvi de link. Així doncs, arribarem fins a la nova unitat amb un KO Link i farem sortir un OK Link des d'aquesta.

- Creem una *No Op Operation Unit* (KO to OK).
    - Unim New Password? amb KO to OK amb un KO Link.
    - Unim KO to OK amb User Modified amb un OK Link.
- Afegim un *Output Collector Parameter* a User Modified amb el nom oidUser que és el paràmetre de sortida de la transacció.
    - Unim l'*Input Collector* amb User Modified amb un Transport Link.
        - Coupling:
            - oidUser amb oidUser.



Il·lustració 40: TxModifyUser. Comprovació i modificació de la contrasenya i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.10 TxRemoveFavorite

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Remove Favorite.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

- oidThread
- oidUser

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

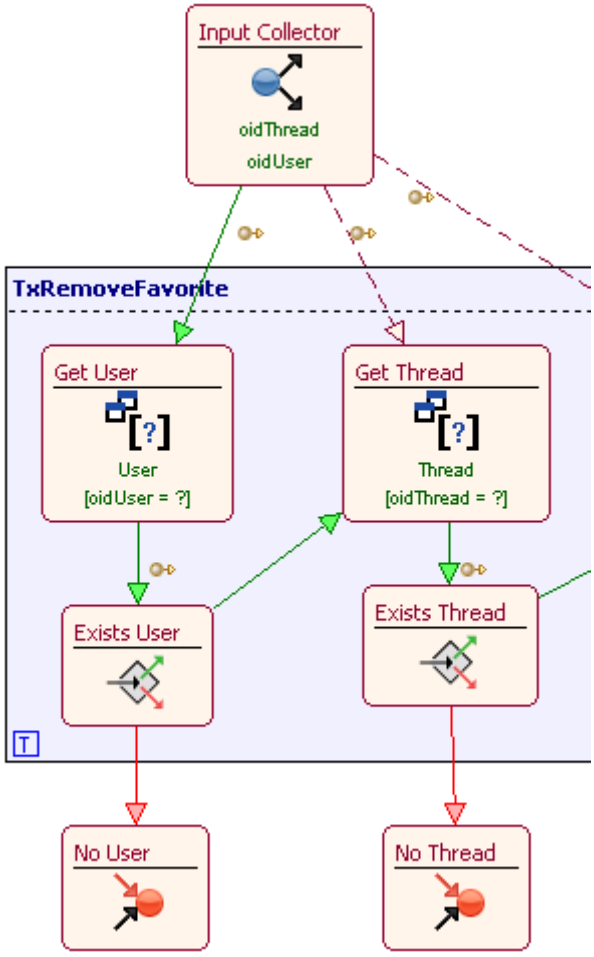
El primer que fem és comprovar els paràmetres. En aquest cas hem de tenir en compte que l'usuari i el tema de discussió siguin vàlids.

A l'UML, hauríem d'agafar les instàncies de l'usuari i del tema de discussió perquè les necessitem a la desfer la relació (eliminant l'usuari de la llista d'usuaris que han fet favorit el tema del tema i al revés). En el cas de WebRatio només necessitem l'identificador de les dues instàncies per a desfer la relació directament ja que és una relació N-N i a nivell de base de dades estem eliminant una tupla d'una sola taula diferent a la de les instàncies. Tot i així, en els dos casos accedirem a la base de dades per a extreure la instància, i com en cas de no existir retorna *null*, haurem de tractar el resultat.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari de la base de dades.
  - Afegim una *Key Condition* a l'entitat User.
  - Unim l'*Input Collector* amb Get User amb un OK Link.
    - Coupling:
      - oidUser amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
  - Unim Get User amb Exists User amb un OK Link.
    - Coupling:
      - oidUser amb l'*Input*.
  - Unim Exists User amb un *KO Collector* que anomenarem No User amb un KO Link.
- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure l'usuari de la base de dades.
  - Afegim una *Key Condition* a l'entitat Thread.
  - Unim Exists User amb Get Thread amb un OK Link.
  - Unim l'*Input Collector* amb Get Thread amb un Transport Link.
    - Coupling:
      - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
  - Unim Get Thread amb Exists Thread amb un OK Link.

- Coupling:
  - oidThread amb l'*Input*.

- Unim Exists Thread amb un *KO Collector* que anomenarem No Thread amb un KO Link.



Il·lustració 41: TxRemoveFavorite. Comprovació de l'usuari i el tema de discussió (WebML)

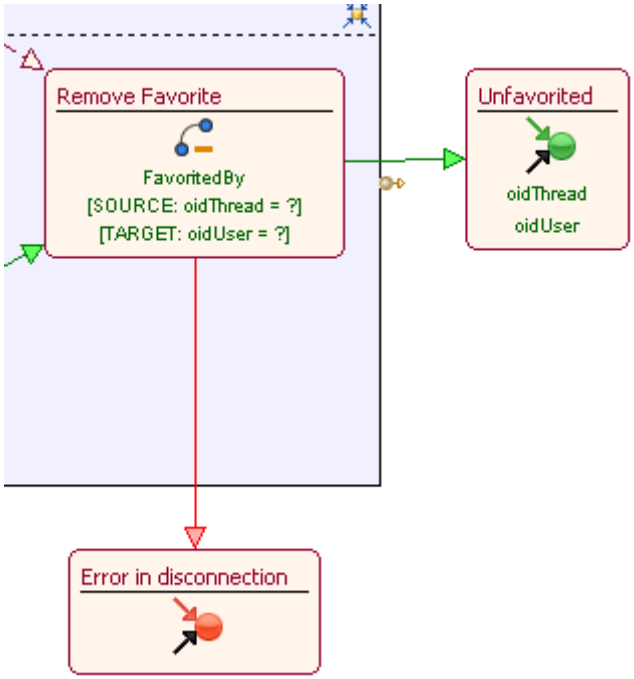
Un cop sabem que els dos identificadors són vàlids utilitzarem una *Disconnect Unit* per a desfer la relació entre les dues instàncies. La *Disconnect Unit* s'utilitza en les relacions N-N del model Entity-Relationship en el que basem l'aplicació. Donada una relació i els dos identificadors (origen i destí) l'elimina.

- Creem una *Disconnect Unit* (Remove Favorite) de la relació FavoritedBy entre Thread i User.
  - Unim Exists Thread amb Remove Favorite amb un OK Link.
  - Unim l'*Input Collector* amb Remove Favorite amb un Transport Link.
    - Coupling:
      - oidThread amb KeyCondition[oidThread][Thread][Source].
      - oidUser amb KeyCondition[oidUser][User][Target].
  - Unim Remove Favorite amb un *OK Collector* que anomenarem Unfavorited amb un OK Link.

- Unim Remove Favorite amb un *KO Collector* que anomenarem Error in disconnection amb un KO Link.

Per a acabar, un cop relacionades les instàncies, modifiquem el *OK Collector* per a poder tenir els paràmetres de sortida de la transacció.

- Afegim un *Output Collector Parameter* a Unfavorited amb el nom oidUser que és un paràmetre de sortida de la transacció.
- Afegim un *Output Collector Parameter* a Unfavorited amb el nom oidThread que és un paràmetre de sortida de la transacció.
  - Unim Remove Favorite amb Unfavorited amb un OK Link.
    - Coupling:
      - Source oidThread [Thread] amb oidThread.
      - Target oidUser [User] amb oidUser.



Il·lustració 42: TxRemoveFavorite. Eliminació de la relació i Collectors (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.11 TxValidateLogin

Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Login.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

- password
- username

A continuació comencem amb les operacions de la transacció. Com ja s'ha comentat abans, utilitzem un *Operation Group* per a indicar que volem un comportament transaccional per a totes les operacions que inclourem a dins. D'aquesta forma si trobem qualsevol error l'execució no crearia inconsistències.

Les condicions per a poder fer el login són senzilles. Simplement necessitem que la parella username – password amb la que es crida la transacció existeixi.

Per tant, accedim a la base de dades per a extreure la instància i comprovar si existeix. Si existeix modificarem la data d'última connexió i sinó es tornara un identificador d'usuari buit en comptes de l'identificador de l'usuari.

Començarem extreient la instància de la que necessitarem també l'identificador per a després modificar-la.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure l'usuari de la base de dades.
    - Afegim una *Attribute Condition* per a l'atribut "password" de l'entitat User que anomenarem "password".
    - Afegim una *Attribute Condition* per a l'atribut "userName" de l'entitat User que anomenarem "username".
    - Unim l'*Input Collector* amb Get Thread amb un OK Link.
        - Coupling:
            - password amb l'*Attribute Condition* "password".
            - userName amb l'*Attribute Condition* "username".
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.

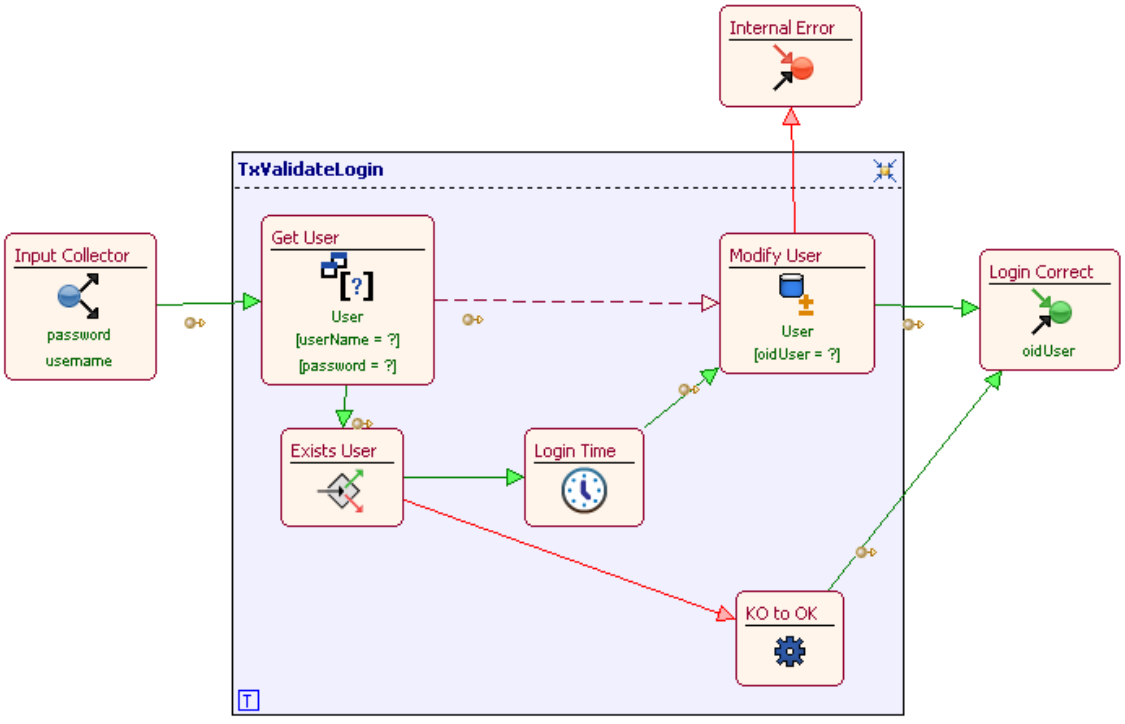
En el cas de que existeixi l'usuari volem fer la modificació de l'últim cop que ha entrat al sistema.

- Creem una *Time Unit* (Login Time) per a accedir al temps actual del sistema.
    - Unim Exists User amb Login Time amb un OK Link.
- Creem una *Modify Unit* (Modify User) de l'entitat User per a poder fer la modificació de l'última connexió.
    - Unim Get User amb Modify User amb un Transport Link.
        - Coupling:

        - oidUser amb KeyCondition[oidUser].
    - Unim Login Time amb Modify User amb un OK Link.
        - Coupling:
            - Current/Input Timestamp amb last_connection.
    - Unim Modify User amb un *OK Collector* que anomenarem Login Correct amb un OK Link.
- Afegim un *Output Collector Parameter* a Login Correct amb el nom oidUser que és el paràmetre de sortida de la transacció.
    - Modifiquem el OK Link entre Modify User i Login Correct
        - Coupling:
            - oidUser amb oidUser.

Per altra banda, si no existeix l'usuari amb el nom i contrasenya entrats, volem arribar a Login Correct sense emparellar cap identificador. Com no podem enllaçar l'Exists User amb Login Correct amb un KO Link perquè els *OK Collector* no accepten KO Links d'entrada, crearem una *No Op Operation Unit* que ens farà de pont permetent que arribem a ella amb un KO Link i en sortim amb un OK Link sense fer cap tipus d'operació.

- Creem una *No Op Operation Unit* (KO to OK).
    - Unim Exists User amb KO to OK amb un KO Link.
    - Unim KO to OK amb Login Correct amb un OK Link.



Il·lustració 43: TxValidateLogin (WebML)

Els *KO Collector* i els *OK Collector* estan col·locats fora de l'*Operation Group*.

7.2.12 TxGetIndex, TxGetUserProfile i TxGetThreadInfo

Aquestes tres transaccions al traduir d'UML a WebRatio desapareixen tal i com les coneixíem.

WebRatio ofereix una sèrie d'unitats anomenades de contingut o *Content Units.* Aquestes unitats estan dins les pàgines i no a fora com ho estan la resta d'unitats que hem estat usant tot i que algunes d'aquestes són híbrides i també poden usar-se d'aquesta manera.

El que tenen en comú aquestes tres transaccions és que són transaccions purament d'extracció d'informació però no de modificació o creació de nova informació per a l'aplicació.

Així doncs el que farem és incorporar aquesta lògica dins de la lògica de la pàgina en comptes de treure-la fora com hem fet amb les transaccions.

Podrem veure cada un dels comportaments d'aquestes transaccions a les següents pàgines:

- TxGetIndex a la pàgina Index.
- TxGetUserInfo a la pàgina User Profile.
- TxGetThreadInfo a la pàgina Thread Info.

Tot i així, TxGetThreadInfo també és usada al generar el formulari de modificació d'un tema de discussió i en aquest cas si que hauríem de tenir aquesta lògica en una transacció. De totes formes, com no ens interessen les seves respostes i només volem certa informació de la instància del tema, la simplifiquem. Es pot veure la seva implementació en el següent apartat.

7.2.13 TxGetThreadInfo

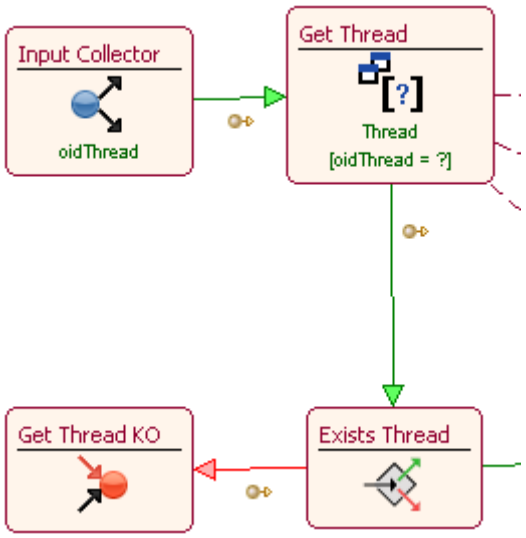
Comencem creant un nou mòdul dins de Forum Transactions. En aquest cas l'anomenarem Get Thread Info.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada que té la transacció.

- oidThread

En aquest cas que nomes estem extraient informació de la base de dades i que no farem cap modificació, no és necessari englobar les operacions en una transacció.

- Creem una Selector Unit (Get Thread) de l'entitat Thread.
  - Afegim una *Key Condition* a l'entitat Thread.
  - Unim l'*Input Collector* amb Get Thread amb un OK Link.
    - Coupling:
      - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
  - Unim Get Thread amb Exists Thread amb un OK Link.
    - Coupling:
      - oidThread amb l'*Input*.
  - Unim Exists Thread amb un *KO Collector* que anomenarem GetThread KO amb un KO Link.



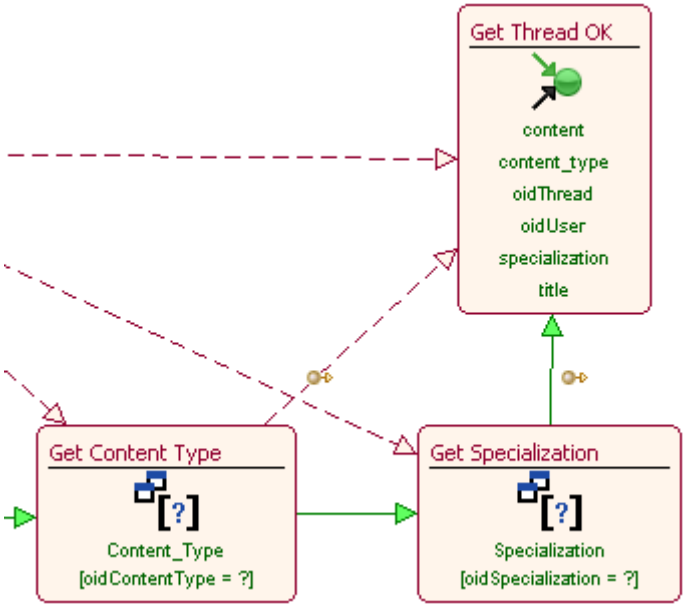
Il·lustració 44: TxGetThreadInfo. Comprovació del tema de discussió (WebML)

- Creem una Selector Unit (Get Content Type) de l'entitat Content_Type.
  - Afegim una *Key Condition* a l'entitat Content_Type.
  - Unim Get Thread amb Get Content Type amb un Transport Link.
    - Coupling:
      - HasType.oidContentType amb la *Key Condition*.
  - Unim Exists Thread amb Get Content Type amb un OK Link.
- Creem una Selector Unit (Get Specializaton) de l'entitat Specialization.

- Afegim una *Key Condition* a l'entitat Thread.
- Unim Get Thread amb Get Thread amb un Transport Link.
  - Coupling:
    - HasSpecialization.oidSpecialization amb la *Key Condition*.
- Unim Exists Thread amb Get Specializaton amb un OK Link.

A partir d'un *OK Collector* retornarem tota la informació necessària de la transacció.

- Creem un *OK Collector* que anomenarem Get Thread OK.
  - Afegim un *Output Collector Parameter* amb el nom content.
  - Afegim un *Output Collector Parameter* amb el nom content_type.
  - Afegim un *Output Collector Parameter* amb el nom oidThread.
  - Afegim un *Output Collector Parameter* amb el nom oidUser.
  - Afegim un *Output Collector Parameter* amb el nom specialization.
  - Afegim un *Output Collector Parameter* amb el nom title.
  - Unim Get Thread amb Get Thread OK amb un Transport Link.
    - Coupling:
      - content amb content.
      - oidThread amb oidThread.
      - PostedBy.oidUser amb oidUser.
      - title amb title.
  - Unim Get Content Type amb Get Thread OK amb un Transport Link.
    - Coupling:
      - type amb content_type.
  - Unim Get Specialization amb Get Thread OK amb un OK Link.
    - Coupling:
      - name amb specialization.



Il·lustració 45: TxGetThreadInfo. Extracció del tipus de contingut i especialitat i Collector (WebML)

7.3 Business Helpers

Com ja s'ha explicat durant l'especificació, els *business helper* són unes classes que interactuen amb la capa de domini per a executar les operacions necessàries per a una funcionalitat o per a extreure informació i a donar-se-la a les *server page* per a renderitzar la pàgina.

A WebRatio hi ha diferents tipus d'unitats que ens permeten extreure informació i algunes permeten també renderitzar-les directament. Per això, a l'hora de fer la traducció de l'aplicació s'ha optat per a separar en dues parts els *business helper* i redistribuir les responsabilitats.

Per una banda, el *business helper* seguirà fent tota la feina de comunicació amb domini i extraurà la informació d'entitats que no s'hagin de mostrar directament.

Per una altra, tota la informació que puguem mostrar directament estarà dins la pàgina o podríem dir que s'ha mogut cap a la *Server Page* en el cas de l'especificació.

És a dir, les *Content Unit* seran exclusives de les pàgines i però no podran utilitzar cap altre tipus d'unitat.

En aquest punt s'explicarà la traducció dels *business helper* amb la modificació que hem explicat ara ja que la part de les pàgines l'explicarem en el següent punt.

Per a fer la interacció amb les transaccions que hem definit com a mòduls, utilitzarem el mateix mètode. Així doncs, crearem un mòdul per a cada un dels *business helper* que hem utilitzat a l'aplicació i generarem el codi tenint en compte el que hem explicat. Aquests mòduls estaran en una nova *Module View* que anomenarem Forum Business Helpers.

A partir dels *OK* i *KO Collectors* podrem donar varies sortides al mòdul per a poder-ho gestionar des de fora en el cas de que vulguem tenir comportaments diferents segons els resultats que obtinguem de les operacions executades.

7.3.1 Add Favorite BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Add Favorite BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidThread

A continuació comencem a modelar el comportament del *business helper*. Principalment el que s'ha de fer és obtenir de la sessió l'usuari actual per a cridar la transacció TxAddFavorite i gestionar els resultats d'aquesta operació segons convingui.

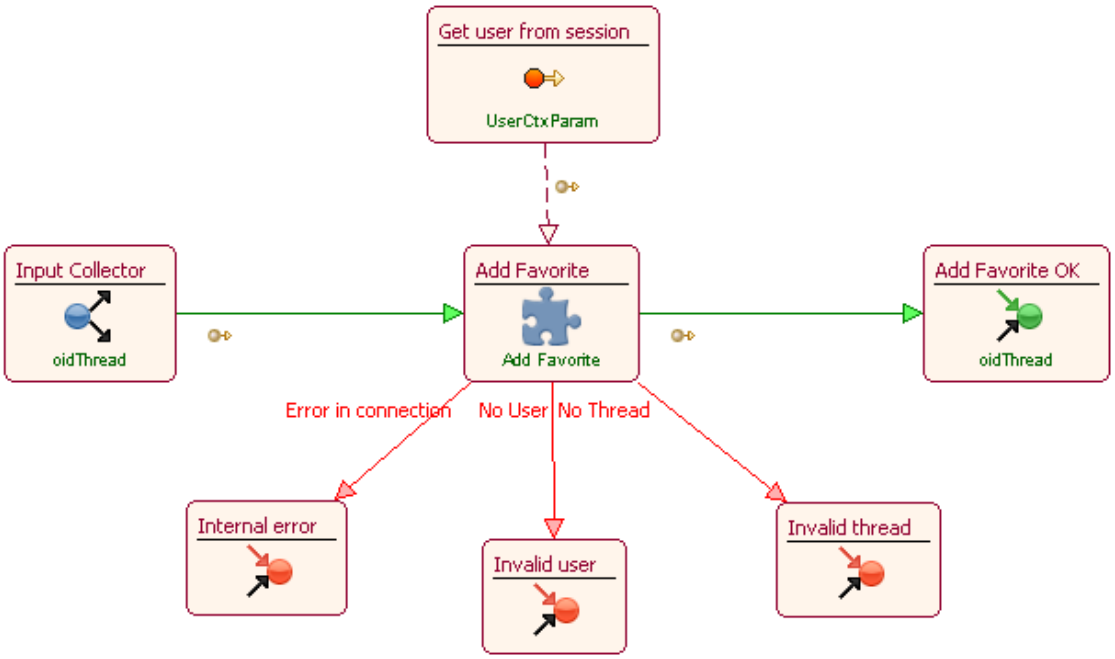
- Creem una *Module Unit* (Add Favorite) del tipus Add Favorite.
    - Unim l'*Input Collector* amb Add Favorite amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb Add Favorite amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb oidUser.

En cas correcte volem retornar l'identificador del tema de discussió.

- Creem un *OK Collector* que anomenarem Add Favorite OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Add Favorite amb Add Favorite OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim tres opcions d'error, una per a cada error del mòdul Add Favorite.

- Creem un *KO Collector* que anomenarem Invalid User.
    - Unim Add Favorite amb Invalid User.
        - Collector Code: No User.
- Creem un *KO Collector* que anomenarem Invalid Thread.
    - Unim Add Favorite amb Invalid Thread.
        - Collector Code: No Thread.
- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Add Favorite amb Internal Error.
        - Collector Code: Error in connection.



Il·lustració 46: Add Favorite BH (WebML)

7.3.2 Create Thread BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Create Thread BH.

Dins del mòdul crearem un *Input Collector*. En aquest cas no necessita tenir cap paràmetre però s'utilitza com a unitat d'entrada al *business helper*.

A continuació comencem a modelar el comportament del *business helper*. En aquest cas volem accedir al llistat de possibles tipus de contingut i especialitats que hi ha disponibles al sistema per a després poder-les mostrar i que així l'usuari pugui escollir-ne una de cada.

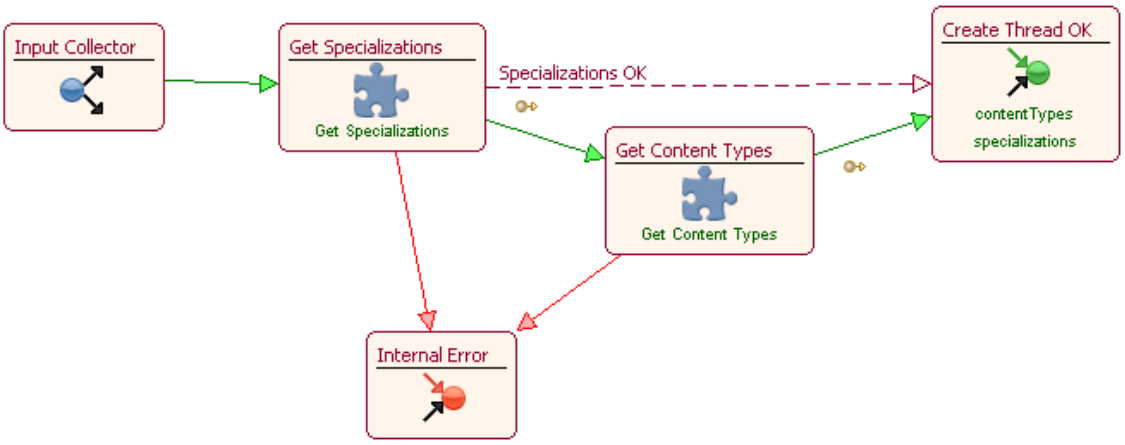
- Creem una *Module Unit* (Get Specializations) del tipus Get Specializations.
  - Unim l'*Input Collector* amb Get Specializations amb un OK Link.
- Creem una *Module Unit* (Get Content Types) del tipus Get Content Types.
  - Unim Get Specializations amb Get Specializations amb un OK Link.

En cas correcte volem retornar les dues llistes.

- Creem un *OK Collector* que anomenarem Create Thread OK.
  - Afegim un *Output Collector Parameter* amb el nom contentTypes.
  - Afegim un *Output Collector Parameter* amb el nom specializations.
  - Unim Get Specializations amb Create Thread OK amb un Transport Link.
    - Collector Code: Specializations OK.
    - Coupling:
      - name amb specializations.
  - Unim Get Content Types amb Create Thread OK amb un OK Link.
    - Coupling:
      - type amb contentTypes.

I en cas d'error només volem un punt de sortida ja que l'únic possible error pels dos mòduls és un fallo intern al sistema.

- Creem un *KO Collector* que anomenarem Internal Error.
  - Unim Get Specializations amb Internal Error amb un KO Link.
  - Unim Get Content Types amb Internal Error amb un KO Link.



Il·lustració 47: Create Thread BH (WebML)

7.3.3 Create User BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Create User BH.

Dins del mòdul crearem un *Input Collector*. En aquest cas no necessita tenir cap paràmetre però s'utilitza com a unitat d'entrada al *business helper*.

A continuació comencem a modelar el comportament del *business helper*. En aquest cas volem accedir al llistat de possibles especialitats que hi ha disponibles al sistema per a després poder-les mostrar i que així l'usuari pugui escollir-ne una.

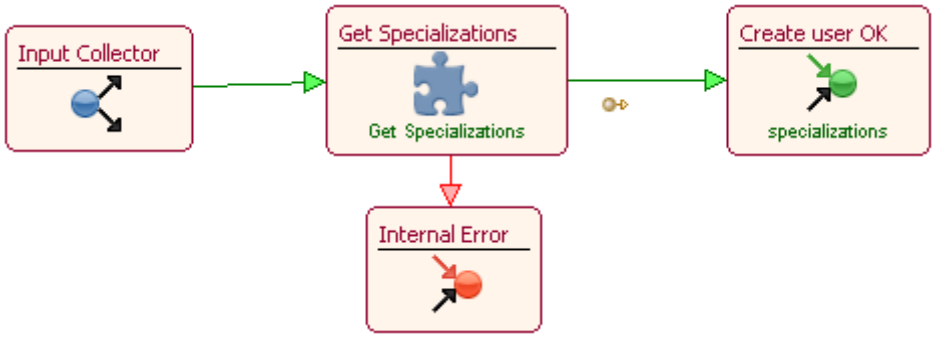
- Creem una *Module Unit* (Get Specializations) del tipus Get Specializations.
    - Unim l'*Input Collector* amb Get Specializations amb un OK Link.

En cas correcte volem retornar la llista.

- Creem un *OK Collector* que anomenarem Create User OK.
    - Afegim un *Output Collector Parameter* amb el nom specializations.
    - Unim Get Specializations amb Create Thread OK amb un Transport Link.
        - Coupling:
            - name amb specializations.

I en cas d'error només volem un punt de sortida ja que l'únic possible error és un fallo intern al sistema.

- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Get Specializations amb Internal Error amb un KO Link.



Il·lustració 48: Create User BH (WebML)

7.3.4 Do Login BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Do Login BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- password
- username

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Login per a cridar la transacció TxValidateLogin per a intentar iniciar sessió al sistema. En cas d'iniciar-la, guardarem l'identificador de l'usuari a la sessió i en cas contrari crearem un missatge d'error per a mostrar-li.

- Creem una *Module Unit* (Login) del tipus Login.
    - Unim l'*Input Collector* amb Login amb un OK Link.
        - Coupling:
            - password amb password.
            - username amb username.
- Creem una *Set Unit* (Set user in session) amb UserCtxParam com a *Context Parameter*.
    - Unim Login amb Set user in session amb un Transport Link.
        - Collector Code: Login Correct.
        - Coupling:
            - oidUser amb UserCtxParam.oidUser.
- Creem un *KO Collector* que anomenarem Internal error.
    - Unim Login amb Internal error amb un KO Link.
        - Collector Code: Internal Error.

Com a resultat de la transacció obtenim l'identificador de l'usuari en cas de que els paràmetres siguin correctes o *null* en el cas de que no. Per tal de comprovar-ho utilitzarem una *Is Not Null Unit*.

- Creem una *Is Not Null Unit* (Validation OK) per a comprovar el valor de retorn de la transacció.
    - Unim Login amb Validation OK amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.

En el cas de tenir l'identificador acabem amb el *business helper*.

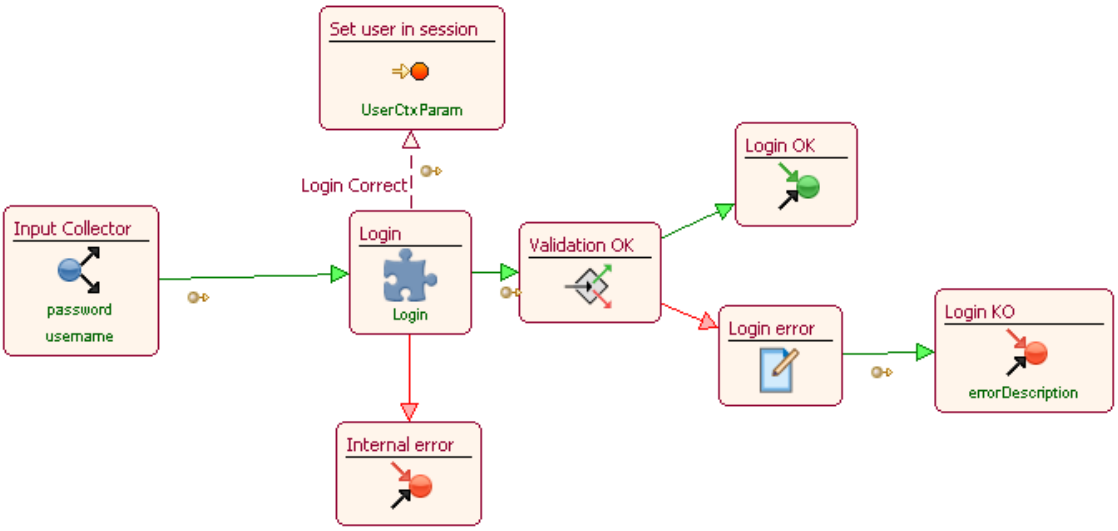
- Creem un *OK Collector* que anomenarem Login OK.
    - Unim Validation OK amb Login OK amb un OK Link.

Si el mòdul de Login ens indica que l'usuari no existeix volem mostrar un missatge per pantalla. Per a fer això utilitzarem una *Script Unit* per a generar un paràmetre de tipus *String* que retornarem amb un *KO Collector*.

- Creem una *Script Unit* (Login error) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Incorrect username or password." ]
```

  - Unim Validation OK amb Login error amb un KO Link.
- Creem un *KO Collector* que anomenarem Login KO.
  - Afegim un *Output Collector Parameter* amb el nom errorDescription.
  - Unim Login Error amb Login KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.



Il·lustració 49: Do Login BH (WebML)

7.3.5 Do Logout BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Do Response Modification BH.

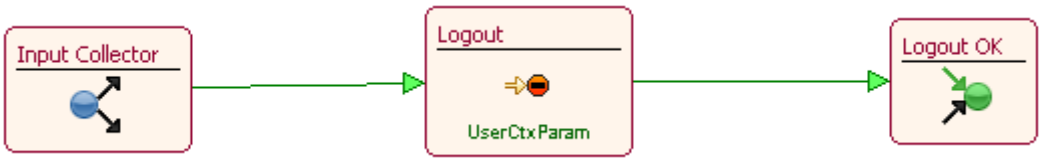
Dins del mòdul crearem un *Input Collector*. En aquest cas no necessita tenir cap paràmetre però s'utilitza com a unitat d'entrada al *business helper*.

A continuació comencem a modelar el comportament del *business helper*. En aquest cas l'únic que ha de fer és les dades de l'usuari de la sessió actual.

Per tal d'eliminar les dades guardades en sessió utilitzarem una *Reset Unit*. Aquesta unitat permet eliminar totes les dades donat un *Context Parameter*.

- Creem una *Reset Unit* (Logout) amb UserCtxParam com a Context *Parameter*.
  - Unim l'*Input Collector* amb Logout amb un OK Link.
- Creem un *OK Collector* que anomenarem Logout OK.
  - Unim Logout amb Logout OK amb un OK Link.

Com de la *Reset Unit* no pot sortir un KO Link, no hi haurà cap error possible.



Il·lustració 50: Do Logout BH (WebML)

7.3.6 Do Response Modification BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Do Response Modification BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- content
- oidResponse

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Modify Response per a cridar la transacció TxModifyResponse per a fer la modificació. La transacció en qüestió només volem que s'executi en el cas de que l'usuari que esta en la sessió sigui l'autor de la resposta.

Per tal de comprovar això utilitzarem una *Math Unit* per a fer la comparació i després tractarem el resultat amb una *Switch Unit* per a fer coses diferents segons és el mateix o no. Cal destacar que la *Math Unit* no accepta valors *null* com a paràmetre així que haurem de comprovar-ho abans.

El primer serà accedir a la base de dades per a extreure l'identificador de l'autor de la resposta tal i com fem amb l'UML.

- Creem una *Selector Unit* (Get Response) de l'entitat Response per a extreure la resposta de la base de dades.
    - Afegim una *Key Condition* a l'entitat Response.
    - Unim l'*Input Collector* amb la unitat Get Response amb un OK Link.
        - Coupling:
            - oidResponse amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Response).
    - Unim Get Response amb Exists Response amb un OK Link.
        - Coupling:
            - oidResponse amb l'*Input*.
- Creem una *Is Not Null Unit* (User in session?) per a saber si existeix l'usuari en sessió.
    - Unim Exists Response amb User in session? amb un OK Link.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb User in session? amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb l'*Input*.

Les *Math Unit* funcionen a partir d'una expressió i operands. L'expressió pot fer servir comparadors o operadors. En aquest cas voldrem comparar dos operands (els identificadors dels usuaris) amb un comparador d'igualtat per a obtenir un resultat booleà.

- Creem una *Math Unit* (Check user) per a fer la comparació dels identificadors.
    - Afegim un *Operand* amb nom Response user.
    - Afegim un *Operand* amb nom Session user.

    - Modifiquem els paràmetres de la unitat.
        - Default Expression: Session user = Response user.
        - Result Type: Boolean.
    - Unim Get Response amb Check user amb un Transport Link.
        - Coupling:
            - FromUser.oidUser amb Response user.
    - Unim Get user from session amb Check user amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb Session user.
    - Unim User in session? amb Check user amb un OK Link.

Un cop tenim el resultat, l'introduirem a una *Switch Unit*. Aquestes unitats tenen uns possibles valors d'entrada i una sortida per a cada valor a més d'un *default*. En definitiva, funcionen com un *switch* en la majoria de llenguatges de programació. D'aquesta manera traduïm la sentència condicional de l'UML fent servir *true* i *false* com a possibles valors per a un booleà.

- Creem una *Switch Unit* (Correct user?) per a comprovar el valor de la Check user.
    - Modifiquem els paràmetres de la unitat.
        - Case Values:
            - true
            - false
    - Unim Check user amb Correct user? amb un OK Link.
        - Coupling:
            - *Result* amb *Switch*.
- Creem una *Module Unit* (Modify Response) del tipus Modify Response.
    - Unim Correct User? amb Modify Response amb un OK Link.
        - Code: true.
    - Unim l'*Input Collector* amb Modify Response amb un Transport Link.
        - Coupling:
            - content amb content.
            - oidResponse amb oidResponse.
- Creem un *OK Collector* que anomenarem Response Modification OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Modify Response amb Response Modification OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim dues opcions d'error. Una és un error intern que no podem controlar i l'altre és un error controlat que volem redirigir amb el missatge d'error corresponent. Per tal de fer els missatges d'error tornarem a fer servir l'*Script Unit*.

- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Modify Response amb Internal error amb un KO Link.
        - Collector Code: Modifying Error.
    - Unim Correct user? amb Internal error amb un OK Link.

        - Code: Cap, en aquest cas un internal error voldrà dir que el resultat no és ni *true* ni *false* per lo que s'ha produït algun error incontrolat.

Les *Script Unit* les utilitzarem en el cas que l'usuari no sigui l'autor o que la resposta a modificar no sigui vàlida.

- Creem una *Script Unit* (Incorrect user) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Only response's user is allowed to modify it." ]
```

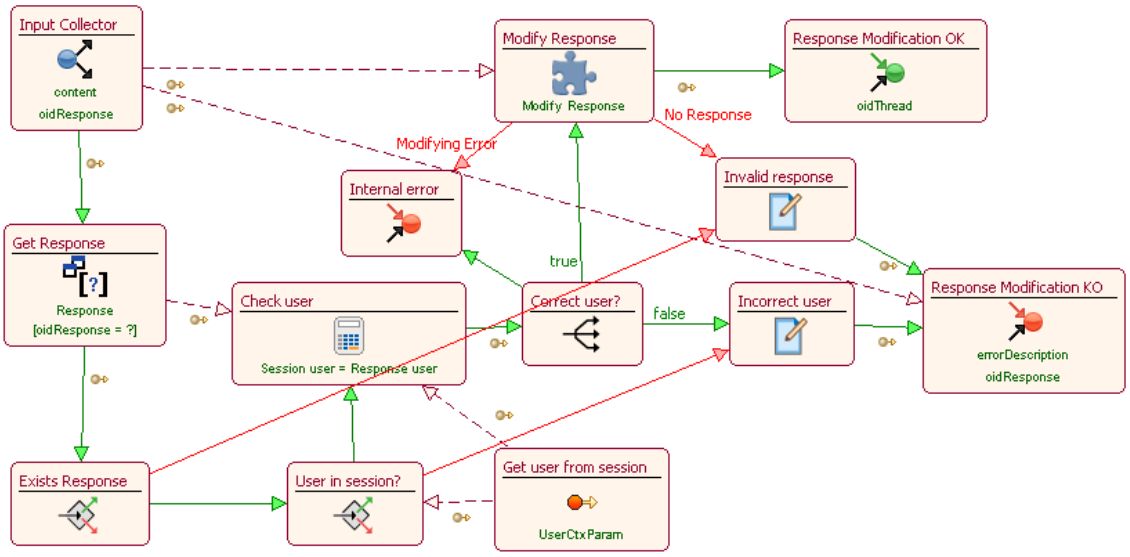
    - Unim Correct user? amb Incorrect user amb un OK Link.
        - Code: false.
    - Unim User in session? amb Incorrect user amb un KO Link.
- Creem una *Script Unit* (Invalid response) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "You are trying to modify an unexisting response." ]
```

    - Unim Modify Response amb Invalid response amb un KO Link.
        - Collector Code: No Response.
    - Unim Exists Response amb Invalid response amb un KO Link.

Com a paràmetres de retorn a l'error, ens interessa tenir el possible error i l'identificador de la resposta ja que mostrarem els errors a la pàgina de la modificació de la resposta.

- Creem un *KO Collector* que anomenarem Response Modification KO.
    - Afegim un *Output Collector Parameter* amb el nom errorDescription.
    - Afegim un *Output Collector Parameter* amb el nom oidResponse.
    - Unim l'*Input Collector* amb Response Modification KO amb un Transport Link.
        - Coupling:
            - oidResponse amb oidResponse.
    - Unim Incorrect user amb Response Modification KO amb un OK Link.
        - Coupling:
            - errorMsg amb errorDescription.
    - Unim Incorrect response amb Response Modification KO amb un OK Link.
        - Coupling:
            - errorMsg amb errorDescription.



Il·lustració 51: Do Response Modification BH (WebML)

7.3.7 Do Thread Modification BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Do Thread Modification BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- content
- content_type
- oidThread
- specialization
- title

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Modify Thread per a cridar la transacció TxModifyThread per a fer la modificació. La transacció en qüestió només volem que s'executi en el cas de que l'usuari que esta en la sessió sigui l'autor del tema de discussió.

Per tal de comprovar això utilitzarem una *Math Unit* per a fer la comparació i després tractarem el resultat amb una *Switch Unit* per a fer coses diferents segons és el mateix o no. Cal destacar que la *Math Unit* no accepta valors *null* com a paràmetre així que haurem de comprovar-ho abans.

El primer serà accedir a la base de dades per a extreure l'identificador de l'autor del tema de discussió tal i com fem amb l'UML.

- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure el tema de discussió de la base de dades.
    - Afegim una *Key Condition* a l'entitat Thread.
    - Unim l'*Input Collector* amb Get Thread amb un OK Link.
        - Coupling:
            - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
    - Unim Get Thread amb Exists Thread amb un OK Link.
        - Coupling:
            - oidThread amb l'*Input*.
- Creem una *Is Not Null Unit* (User in session?) per a saber si existeix l'usuari en sessió.
    - Unim Exists Response amb User in session? amb un OK Link.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb User in session? amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb l'*Input*.

Les *Math Unit* funcionen a partir d'una expressió i operands. L'expressió pot fer servir comparadors o operadors. En aquest cas voldrem comparar dos operands (els identificadors dels usuaris) amb un comparador d'igualtat per a obtenir un resultat booleà.

- Creem una *Math Unit* (Correct user) per a fer la comparació dels identificadors.
    - Afegim un *Operand* amb nom Thread user.
    - Afegim un *Operand* amb nom Session user.
    - Modifiquem els paràmetres de la unitat.
        - Default Expression: Session user = Thread user.
        - Result Type: Boolean.
    - Unim Get Thread amb Correct user amb un Transport Link.
        - Coupling:
            - FromUser.oidUser amb Thread user.
    - Unim Get user from session amb Correct user amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb Session user.
    - Unim User in session? amb Correct user amb un OK Link.

Un cop tenim el resultat, l'introduirem a una *Switch Unit*. Aquestes unitats tenen uns possibles valors d'entrada i una sortida per a cada valor a més d'un *default*. En definitiva, funcionen com un *switch* en la majoria de llenguatges de programació. D'aquesta manera traduïm la sentència condicional de l'UML fent servir *true* i *false* com a possibles valors per a un booleà.

- Creem una *Switch Unit* (Allowed Modify?) per a comprovar el valor de la Correct user.
    - Modifiquem els paràmetres de la unitat.
        - Case Values:
            - true
            - false
    - Unim Correct user amb Allowed Modify? Amb un OK Link.
        - Coupling:
            - *Result* amb *Switch*.
- Creem una *Module Unit* (Modify Thread) del tipus Modify Thread.
    - Unim Allowed Modify? Amb Modify Thread amb un OK Link.
        - Code: true.
    - Unim l'*Input Collector* amb Modify Thread amb un Transport Link.
        - Coupling:
            - content amb content.
            - content_type amb content_type.
            - oidThread amb oidThread.
            - specialization amb specialization.
            - title amb title.
- Creem un *OK Collector* que anomenarem Thread Modification OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Modify Thread amb Thread Modification OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim tres opcions d'error. Una és un error intern que no podem controlar, la segona que el tema de discussió no sigui correcte i l'altre és un error

controlat en els paràmetres o en l'usuari i que volem redirigir amb el missatge d'error corresponent. Per tal de fer els missatges d'error tornarem a fer servir l'*Script Unit*.

- Creem un *KO Collector* que anomenarem Internal Error.
  - Unim Modify Thread amb Internal error amb un KO Link.
    - Collector Code: Error Modifying.
  - Unim Allowed Modify? Amb Internal error amb un OK Link.
    - Code: Cap, en aquest cas un internal error voldrà dir que el resultat no és ni *true* ni *false* per lo que s'ha produït algun error incontrolat.
- Creem un *KO Collector* que anomenarem Invalid thread.
  - Unim Modify Thread amb Invalid thread amb un KO Link.
    - Collector Code: No Thread.
  - Unim Exists Thread amb Invalid thread amb un KO Link.

Les *Script Unit* les utiltizarem en el cas que l'usuari no sigui l'autor o que el tipus de contingut o l'especialitat no siguin vàlides.

- Creem una *Script Unit* (Incorrect user) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Only thread's autor is allowed to modify
the thread." ]
```

  - Unim Allowed Modify? Amb Incorrect user amb un OK Link.
    - Code: false.
  - Unim User in session? amb Incorrect user amb un KO Link.
- Creem una *Script Unit* (Invalid content type) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Invalid content type." ]
```

  - Unim Modify Response amb Invalid response amb un KO Link.
    - Collector Code: No Content Type.
- Creem una *Script Unit* (Invalid specialization) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Invalid specialization." ]
```

  - Unim Modify Response amb Invalid response amb un KO Link.
    - Collector Code: No Content Type.

Unim les tres unitats amb el *KO Collector* que amb el que controlarem el missatge d'error.

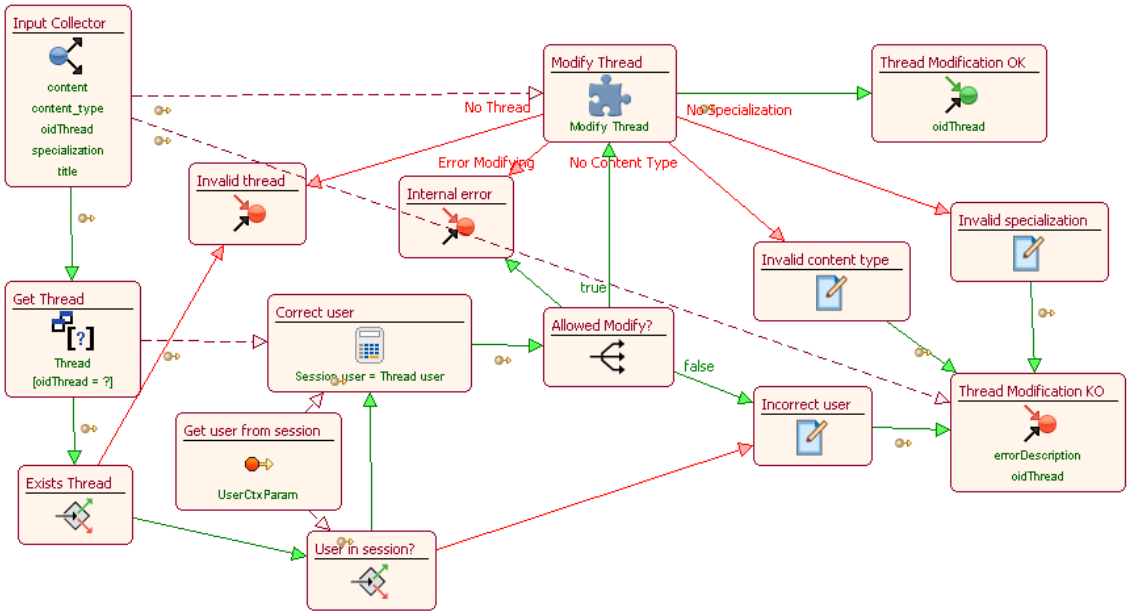
- Creem un *KO Collector* que anomenarem Thread Modification KO.
  - Afegim un *Output Collector Parameter* amb el nom errorDescription.
  - Afegim un *Output Collector Parameter* amb el nom oidThread.
  - Unim l'*Input Collector* amb Thread Modification KO amb un Transport Link.
    - Coupling:
      - oidThread amb oidThread.
  - Unim Incorrect user amb Thread Modification KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Incorrect content type amb Thread Modification KO amb un OK Link.
    - Coupling:

- errorMsg amb errorDescription.

○ Unim Incorrect specialization amb Thread Modification KO amb un OK Link.

▪ Coupling:

- errorMsg amb errorDescription.



Il·lustració 52: Do Thread Modification BH (WebML)

7.3.8 Do User Modification BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Do User Modification BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- email
- newPassword
- oidUser
- oldPassword

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Modify User per a cridar la transacció TxModifyUser per a fer la modificació. La transacció en qüestió només volem que s'executi en el cas de que l'usuari que esta en la sessió sigui l'usuari a modificar.

Per tal de comprovar això utilitzarem una *Math Unit* per a fer la comparació i després tractarem el resultat amb una *Switch Unit* per a fer coses diferents segons és el mateix o no. Cal destacar que la *Math Unit* no accepta valors *null* com a paràmetre així que haurem de comprovar-ho abans.

- Creem una *Is Not Null Unit* (User passed?) per a comprovar que el paràmetre de l'usuari no és *null*.
    - Unim l'*Input Collector* amb User passed? amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context parameter*.
- Creem una *Is Not Null Unit* (User in session?) per a comprovar que hi ha algun usuari a la sessió actual.
    - Unim Get user from session amb User in session? amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb l'*Input*.
    - Unim User passed? amb User in session? amb un OK Link.

Les *Math Unit* funcionen a partir d'una expressió i operands. L'expressió pot fer servir comparadors o operadors. En aquest cas voldrem comparar dos operands (els identificadors dels usuaris) amb un comparador d'igualtat per a obtenir un resultat booleà.

- Creem una *Math Unit* (Correct user) per a fer la comparació dels identificadors.
    - Afegim un *Operand* amb nom Modifying user.
    - Afegim un *Operand* amb nom Session user.
    - Modifiquem els paràmetres de la unitat.
        - Default Expression: Modifying user = Session user.
        - Result Type: Boolean.
    - Unim l'*Input Collector* amb Correct user amb un Transport Link.
        - Coupling:

            - oidUser amb Modifying user.
        - Unim Get user from session amb Correct user amb un Transport Link.
            - Coupling:
                - UserCtxParam.oidUser amb Session user.
        - Unim User in session? amb Correct user amb un OK Link.

Un cop tenim el resultat, l'introduirem a una *Switch Unit*. Aquestes unitats tenen uns possibles valors d'entrada i una sortida per a cada valor a més d'un *default*. En definitiva, funcionen com un *switch* en la majoria de llenguatges de programació. D'aquesta manera traduïm la sentència condicional de l'UML fent servir *true* i *false* com a possibles valors per a un booleà.

- Creem una *Switch Unit* (Comparison result) per a comprovar el valor de la Correct user.
    - Modifiquem els paràmetres de la unitat.
        - Case Values:
            - true
            - false
    - Unim Correct user amb Comparison result amb un OK Link.
        - Coupling:
            - *Result* amb *Switch*.
- Creem una *Module Unit* (Modify User) del tipus Modify User.
    - Unim Comparison result amb Modify User amb un OK Link.
        - Code: true.
    - Unim l'*Input Collector* amb Modify User amb un Transport Link.
        - Coupling:
            - email amb email.
            - newPassword amb newPassword.
            - oldPassword amb oldPassword.
    - Unim Get user from session amb Modify User amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb oidUser.
- Creem un *OK Collector* que anomenarem User Modification OK.
    - Afegim un *Output Collector Parameter* amb el nom oidUser.
    - Unim Modify Thread amb Thread Modification OK amb un OK Link.
        - Coupling:
            - oidUser amb oidUser.

En el cas d'aquest *Business Helper* tenim dues opcions d'error. Una és un error intern que no podem controlar i l'altre és un error controlat en els paràmetres o en l'usuari i que volem redirigir amb el missatge d'error corresponent. Per tal de fer els missatges d'error tornarem a fer servir l'*Script Unit*.

- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Modify User amb Internal error amb un KO Link.
        - Collector Code: Modifying Error.
    - Unim Comparison result amb Internal error amb un OK Link.

        - Code: Cap, en aquest cas un internal error voldrà dir que el resultat no és ni *true* ni *false* per lo que s'ha produït algun error incontrolat.

Les *Script Unit* les utiltizarem en el cas que l'usuari no sigui l'autor o que el tipus de contingut o l'especialitat no siguin vàlides.

- Creem una *Script Unit* (Incorrect user) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "You can only modify your own user. If you are not logged in, log in now." ]
```

    - Unim Comparisonn result amb Incorrect user amb un OK Link.
        - Code: false.
    - Unim User passed? amb Incorrect user amb un KO Link.
- Creem una *Script Unit* (Incorrect password) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Your 99etorn99 password is incorrect." ]
```

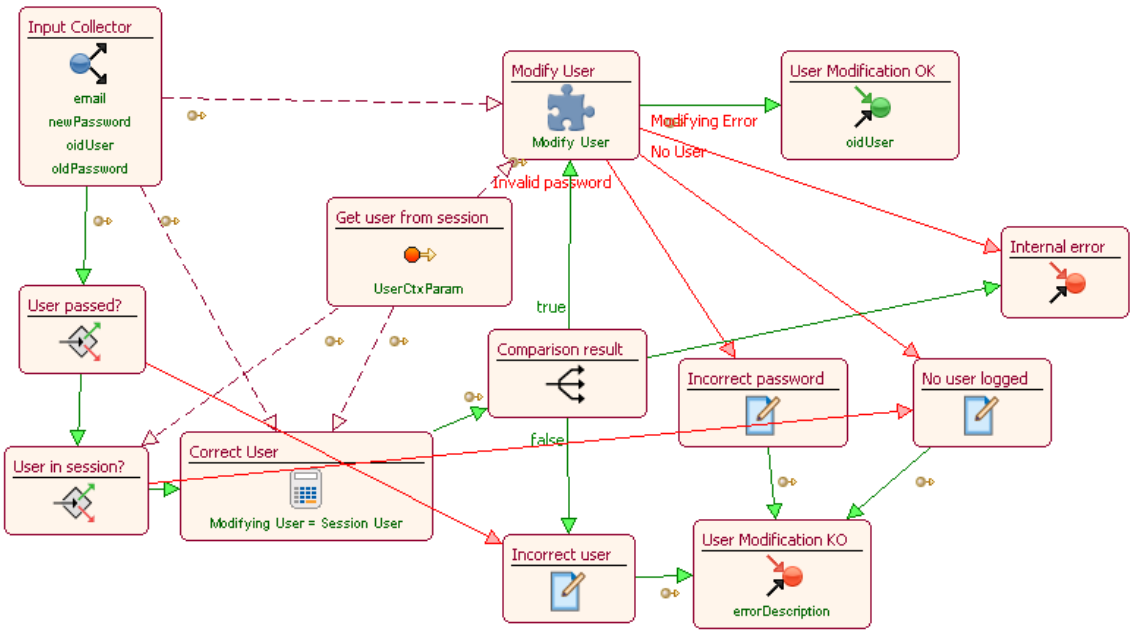
    - Unim Modify User amb Invalid response amb un KO Link.
        - Collector Code: Invalid password.
- Creem una *Script Unit* (No user logged) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "No user is logged in the application. Try again when logged in." ]
```

    - Unim Modify User amb Invalid response amb un KO Link.
        - Collector Code: No User.

Unim les tres unitats amb el *KO Collector* que amb el que controlarem el missatge d'error.

- Creem un *KO Collector* que anomenarem User Modification KO.
    - Afegim un *Output Collector Parameter* amb el nom errorDescription.
    - Unim Incorrect user amb User Modification KO amb un OK Link.
        - Coupling:
            - errorMsg amb errorDescription.
    - Unim Incorrect password amb User Modification KO amb un OK Link.
        - Coupling:
            - errorMsg amb errorDescription.
    - Unim No user logged amb User Modification KO amb un OK Link.
        - Coupling:
            - errorMsg amb errorDescription.



Il·lustració 53: Do User Modification BH (WebML)

7.3.9 Index BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Index BH.

Dins del mòdul crearem un *Input Collector*. En aquest cas no necessita tenir cap paràmetre però s'utilitza com a unitat d'entrada al *business helper*.

A continuació comencem a modelar el comportament del *business helper*. En aquest cas volem accedir al llistat de possibles tipus de contingut i especialitats que hi ha disponibles al sistema per a després poder-les mostrar al formulari de filtre dels temes de discussió. A més a més, necessitem accedir a l'usuari de la sessió per a poder mostrar els seus temes de discussió preferits.

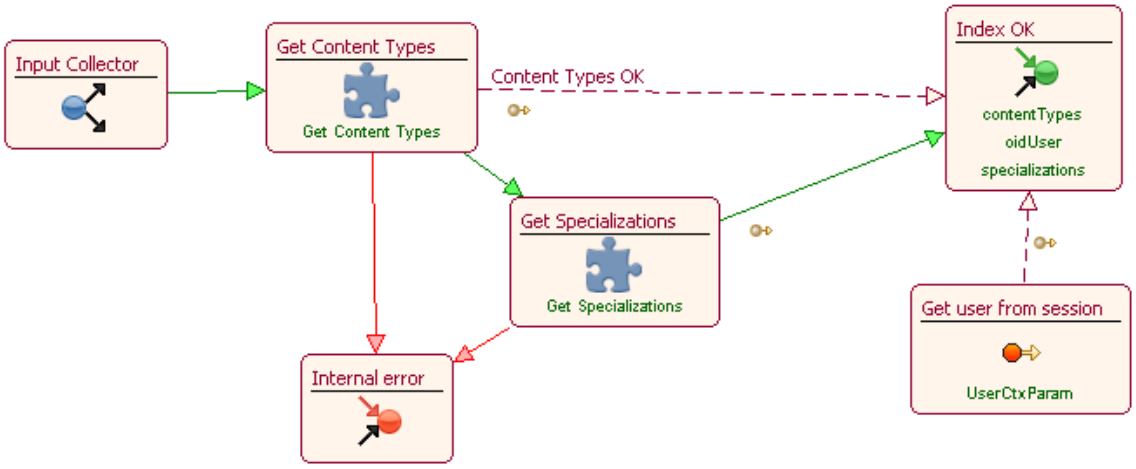
- Creem una *Module Unit* (Get Specializations) del tipus Get Specializations.
    - Unim l'*Input Collector* amb Get Specializations amb un OK Link.
- Creem una *Module Unit* (Get Content Types) del tipus Get Content Types.
    - Unim Get Specializations amb Get Specializations amb un OK Link.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.

En cas correcte volem retornar les dues llistes i l'usuari de la sessió.

- Creem un *OK Collector* que anomenarem Index OK.
    - Afegim un *Output Collector Parameter* amb el nom contentTypes.
    - Afegim un *Output Collector Parameter* amb el nom specializations.
    - Afegim un *Output Collector Parameter* amb el nom oidUser.
    - Unim Get Specializations amb Index OK amb un Transport Link.
        - Coupling:
            - name amb specializations.
    - Unim Get Content Types amb Index OK amb un OK Link.
        - Collector Code: Content Types OK.
        - Coupling:
            - type amb contentTypes.
    - Unim Get user from session amb Index OK

I en cas d'error només volem un punt de sortida ja que l'únic possible error pels dos mòduls és un fallo intern al sistema.

- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Get Content Types amb Internal Error amb un KO Link.
    - Unim Get Specializations amb Internal Error amb un KO Link.



Il·lustració 54: Index BH (WebML)

7.3.10 Modify Response BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Modify Response BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidResponse

A continuació comencem a modelar el comportament del *business helper*. En aquest cas no ens caldrà utilitzar cap transacció de la capa de domini però haurem de consultar la base de dades per a comprovar si l'autor de la resposta que s'ha demanat per a modificar és l'usuari de la sessió o no.

Per tal de comprovar això utilitzarem una *Math Unit* per a fer la comparació i després tractarem el resultat amb una *Switch Unit* per a fer coses diferents segons és el mateix o no. Cal destacar que la *Math Unit* no accepta valors *null* com a paràmetre així que haurem de comprovar-ho abans.

Primer accedim a la base de dades per a obtenir la informació de la resposta. Hem de vigilar que el resultat de la consulta no sigui *null*.

- Creem una *Selector Unit* (Get Response) de l'entitat Response per a extreure la resposta de la base de dades.
    - Afegim una *Key Condition* a l'entitat Response.
    - Unim l'*Input Collector* amb la unitat Get Response amb un OK Link.
        - Coupling:
            - oidResponse amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Response).
    - Unim Get Response amb Exists Response amb un OK Link.
        - Coupling:
            - oidResponse amb l'*Input*.

Comprovem que la resposta té un autor i que hi ha un usuari a la sessió abans de comprovar que siguin el mateix.

- Creem una *Is Not Null Unit* (Has author?) per a comprovar si la resposta té un autor.
    - Unim Get Response amb Has author? amb un Transport Link.
        - Coupling:
            - FromUser.oidUser amb l'*Input*.
    - Unim Exists Response amb Has author? amb un OK Link.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Contex Parameter*.
- Creem una *Is Not Null Unit* (User in session?) per a comprovar si hi ha un usuari a la sessió actual.
    - Unim Get user from session amb User in session? amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb l'*Input*.

    - Unim Has author? amb User in session? amb un OK Link.

Un cop comprovats els paràmetres, utilitzarem la *Math Unit*. Les *Math Unit* funcionen a partir d'una expressió i operands. L'expressió pot fer servir comparadors o operadors. En aquest cas voldrem comparar dos operands (els identificadors dels usuaris) amb un comparador d'igualtat per a obtenir un resultat booleà.

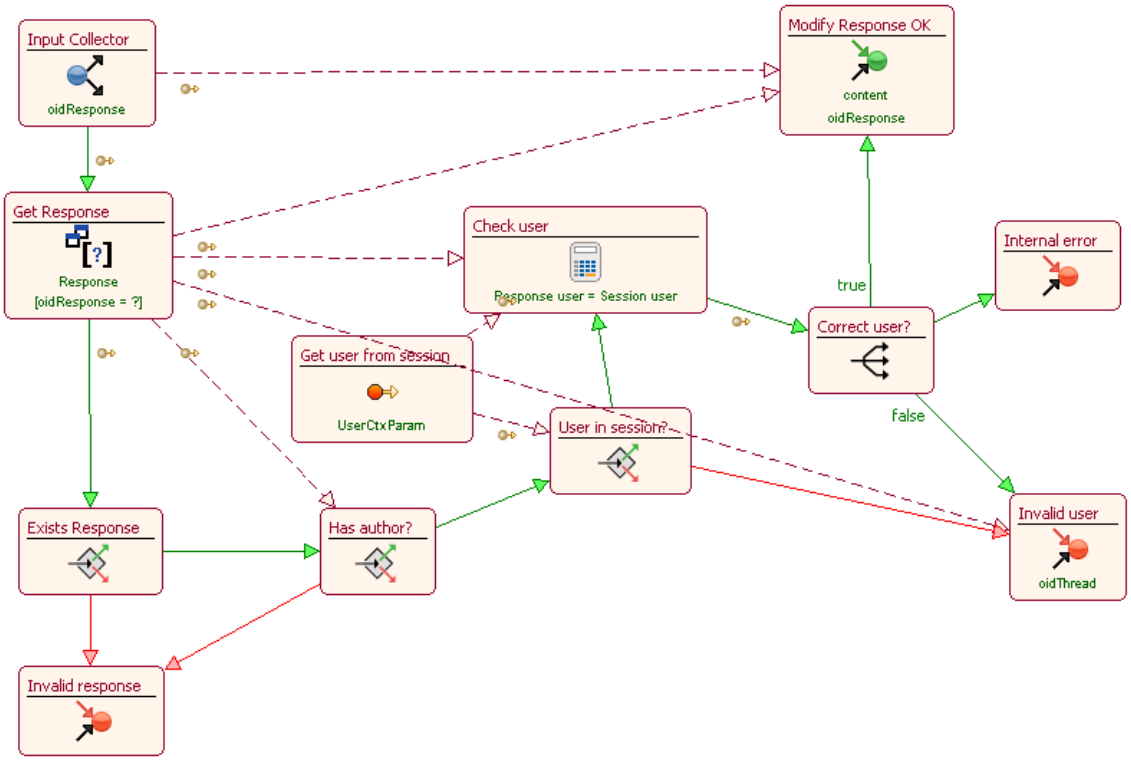
- Creem una *Math Unit* (Check user) per a fer la comparació dels identificadors.
    - Afegim un *Operand* amb nom Response user.
    - Afegim un *Operand* amb nom Session user.
    - Modifiquem els paràmetres de la unitat.
        - Default Expression: Response user = Session user.
        - Result Type: Boolean.
    - Unim Get Response amb Check user amb un Transport Link.
        - Coupling:
            - FromUser.oidUser amb Response user.
    - Unim Get user from session amb Check user amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb Session user.
    - Unim User in session? amb Check user amb un OK Link.

Un cop tenim el resultat, l'introduirem a una *Switch Unit*. Aquestes unitats tenen uns possibles valors d'entrada i una sortida per a cada valor a més d'un *default*. En definitiva, funcionen com un *switch* en la majoria de llenguatges de programació. D'aquesta manera traduïm la sentència condicional de l'UML fent servir *true* i *false* com a possibles valors per a un booleà.

- Creem una *Switch Unit* (Correct user?) per a comprovar el valor de la Check user.
    - Modifiquem els paràmetres de la unitat.
        - Case Values:
            - true
            - false
    - Unim Check user amb Correct user? amb un OK Link.
        - Coupling:
            - *Result* amb *Switch*.
- Creem un *OK Collector* que anomenarem Modify Response OK.
    - Afegim un *Output Collector Parameter* amb el nom content.
    - Afegim un *Output Collector Parameter* amb el nom oidResponse.
    - Unim l'*Input Collector* amb Modify Response OK amb un Transport Link.
        - Coupling:
            - oidResponse amb oidResponse.
    - Unim Get Resonse amb Modify Response OK amb un Transport Link.
        - Coupling:
            - content amb content.
    - Unim Correct user amb Modify Response OK amb un OK Link.
        - Code: true.

En el cas d'aquest *Business Helper* tenim tres opcions d'error. Una és un error intern que no podem controlar, una altra que la resposta no sigui correcte i la tercera és que els usuaris no són iguals o no hi ha ningú a la sessió actual.

- Creem un *KO Collector* que anomenarem Internal Error.
  - Unim Correct user? amb Internal error amb un OK Link.
    - Code: Cap, en aquest cas un internal error voldrà dir que el resultat no és ni *true* ni *false* per lo que s'ha produït algun error incontrolat.
- Creem un *KO Collector* que anomenarem Invalid response.
  - Unim Exists Response amb Invalid response amb un KO Link.
  - Unim Has author? amb Invalid response amb un KO Link.
- Creem un *KO Collector* que anomenarem Invalid user.
  - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Get Response amb Invalid user amb un Transport Link.
      - Coupling:
        - FromThread.oidThread amb oidThread.
  - Unim User in session? amb Invalid user amb un KO Link.
  - Unim Correct user? amb Invalid user amb un OK Link.
    - Code: false.



Il·lustració 55: Modify Response BH (WebML)

7.3.11 Modify Thread BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Modify Thread BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidThread

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem varies transaccions de la capa de domini per tal d'extreure la informació necessària. Els mòduls que s'utilitzaran són Get Content Types, Get Specializations i Get Thread Info, la versió reduïda de l'original ja que no necessitem tota la informació del tema, només aquella que l'usuari podrà modificar.

- Creem una *Module Unit* (Get Content Types) del tipus Get Content Types.
  - Unim l'*Input Collector* amb Get Content Types amb un OK Link.
- Creem una *Module Unit* (Get Specializations) del tipus Get Specializations.
  - Unim Get Content Types amb Get Specializations amb un OK Link.
- Creem una *Module Unit* (Get Thread Info) del tipus Get Thread Info.
  - Unim Get Specializations amb Get Thread Info amb un OK Link.

Abans de poder donar per vàlida l'operació, s'ha de comprovar que l'usuari té permisos per a fer els canvis. És a dir, si l'autor del tema de discussió a modificar és l'usuari de la sessió o no.

Per tal de comprovar això utilitzarem una *Math Unit* per a fer la comparació i després tractarem el resultat amb una *Switch Unit* per a fer coses diferents segons és el mateix o no. Cal destacar que la *Math Unit* no accepta valors *null* com a paràmetre així que haurem de comprovar-ho abans.

- Creem una *Is Not Null Unit* (Has author?) per a comprovar si el tema de discussió té un autor.
  - Unim Get Thread Info amb Has author? amb un OK Link.
    - Coupling:
      - oidUser amb l'*Input*.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Contex Parameter*.
- Creem una *Is Not Null Unit* (User in session?) per a comprovar si hi ha un usuari a la sessió actual.
  - Unim Get user from session amb User in session? amb un Transport Link.
    - Coupling:
      - UserCtxParam.oidUser amb l'*Input*.
  - Unim Has author? amb User in session? amb un OK Link.
- Creem una *Math Unit* (Correct user) per a fer la comparació dels identificadors.
  - Afegim un *Operand* amb nom Thread user.
  - Afegim un *Operand* amb nom Session user.
  - Modifiquem els paràmetres de la unitat.
    - Default Expression: Session user = Thread user.

        - Result Type: Boolean.
    - Unim Get Thread Info amb Correct user amb un Transport Link.
        - Collector Code: Get Thread OK
        - Coupling:
            - oidUser amb Thread user.
    - Unim Get user from session amb Correct user amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb Session user.
    - Unim User in session? amb Correct user amb un OK Link.

Un cop tenim el resultat, l'introduirem a una *Switch Unit*. Aquestes unitats tenen uns possibles valors d'entrada i una sortida per a cada valor a més d'un *default*. En definitiva, funcionen com un *switch* en la majoria de llenguatges de programació. D'aquesta manera traduïm la sentència condicional de l'UML fent servir *true* i *false* com a possibles valors per a un booleà.

- Creem una *Switch Unit* (Equal?) per a comprovar el valor de la Correct user.
    - Modifiquem els paràmetres de la unitat.
        - Case Values:
            - true
            - false
    - Unim Correct user amb Equal? Amb un OK Link.
        - Coupling:
            - *Result* amb *Switch*.

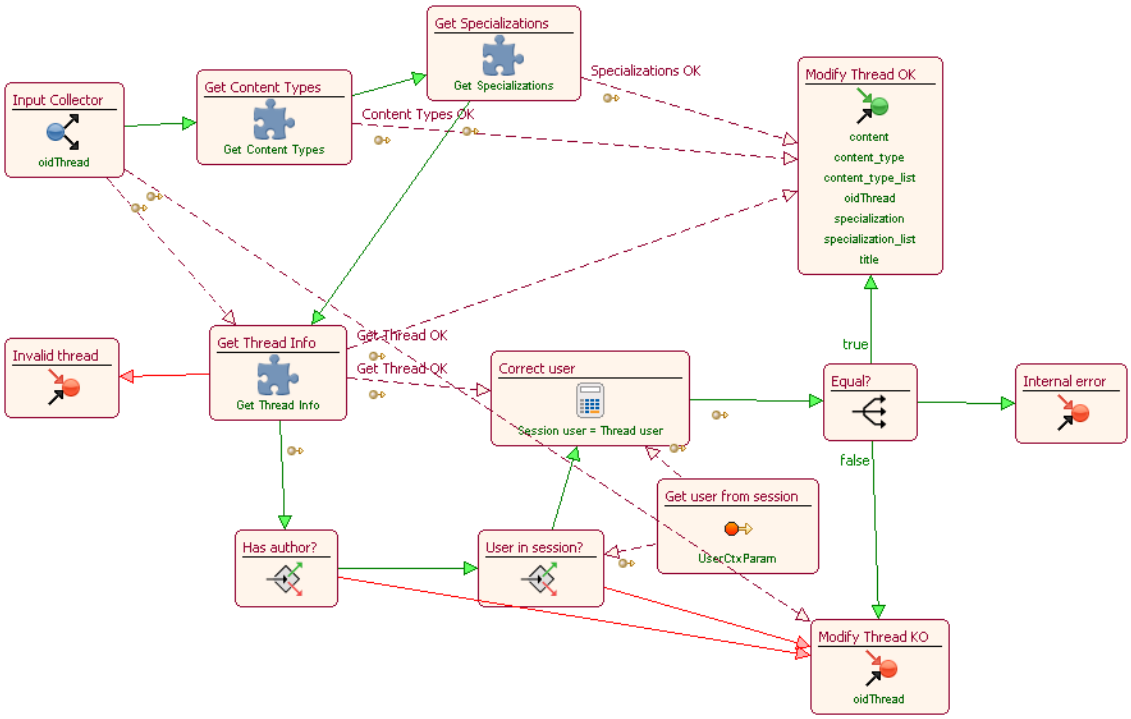
En el cas d'aquesta operació tenim molta informació a extreure de la base de dades per tal de poder omplir correctament el formulari. Haurem de crear l'*OK Collector* i tenir un *Output Collector Parameter* per a cada paràmetre de sortida i omplir tots els valors des de les diferentes unitats que han generat el valor.

- Creem un *OK Collector* que anomenarem Modify Thread OK.
    - Afegim un *Output Collector Parameter* amb el nom content.
    - Afegim un *Output Collector Parameter* amb el nom content_type.
    - Afegim un *Output Collector Parameter* amb el nom content_type_list.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Afegim un *Output Collector Parameter* amb el nom specialization.
    - Afegim un *Output Collector Parameter* amb el nom specialization_list.
    - Afegim un *Output Collector Parameter* amb el nom title.
    - Unim Get Content Types amb Modify Thread OK amb un Transport Link.
        - Collector Code: Content Types OK.
        - Coupling:
            - contentTypes amb content_type_list.
    - Unim Get Specializations amb Modify Thread OK amb un Transport Link.
        - Collector Code: Specializations OK.
        - Coupling:
            - Specializations amb specializations_list.
    - Unim Get Thread Info amb Modify Thread OK amb un Transport Link.

        - Collector Code: Get Thread OK.
        - Coupling:
            - content amb content.
            - content_type amb content_type.
            - oidThread amb oidThread.
            - specialization amb specialization.
            - title amb title.
    - Unim Equal? amb Modify Thread OK amb un OK Link.
        - Code: *true*.

En el cas d'aquest *Business Helper* tenim tres opcions d'error. Una és un error intern que no podem controlar, la segona que el tema de discussió no sigui correcte i l'altre és un error controlat en l'usuari.

- Creem un *KO Collector* que anomenarem Internal error.
    - Unim Equal? amb Internal error amb un OK Link.
        - Code: Cap, en aquest cas un internal error voldrà dir que el resultat no és ni *true* ni *false* per lo que s'ha produït algun error incontrolat.
- Creem un *KO Collector* que anomenarem Invalid thread.
    - Unim Get Thread Info amb Invalid Thread amb un KO Link.
- Creem un *KO Collector* que anomenarem Modify Thread KO.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim l'*Input Collector* amb Modify Thread KO amb un Transport Link.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Has author? amb Modify Thread KO amb un KO Link.
    - Unim User in session? amb Modify Thread KO amb un KO Link.
    - Unim Equal? amb Modify Thread KO amb un OK Link.
        - Code: *false*.



Il·lustració 56: Modify Thread BH (WebML)

7.3.12 Modify User BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Modify User BH.

Dins del mòdul crearem un *Input Collector*. En aquest cas no necessita tenir cap paràmetre però s'utilitza com a unitat d'entrada al *business helper*.

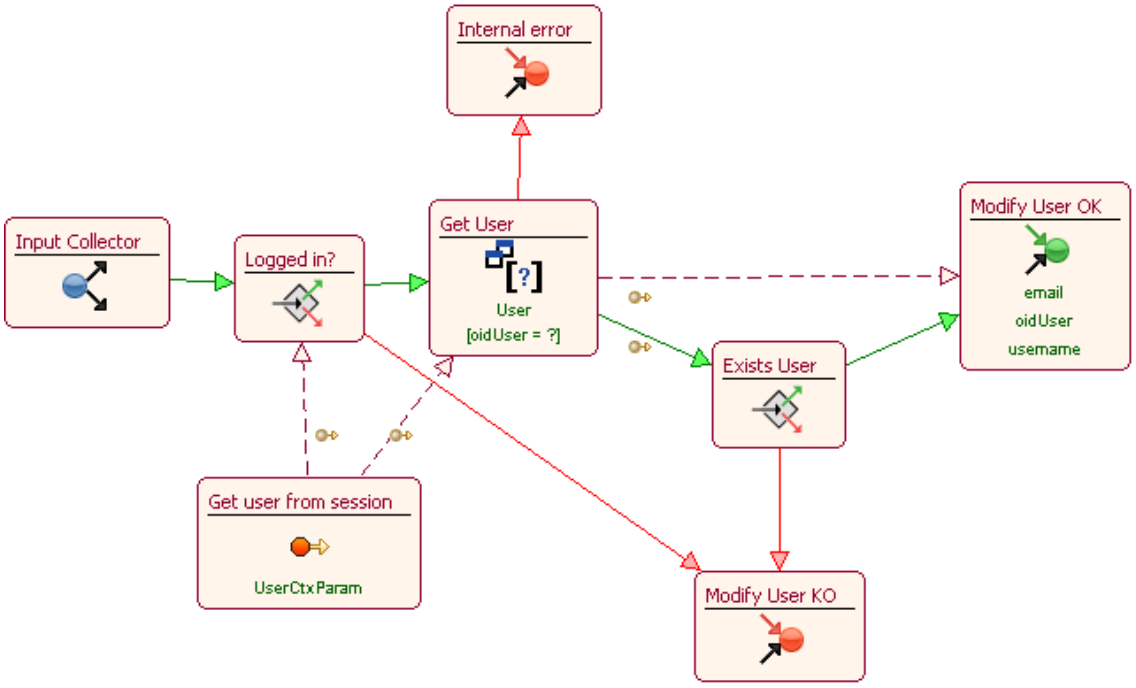
A continuació comencem a modelar el comportament *business helper*. En aquest cas no ens caldrà utilitzar cap transacció de la capa de domini però haurem de consultar la base de dades per a comprovar agafar la informació de l'usuari per a poder omplir els camps per a la modificació. L'únic requisit és que hi hagi un usuari a la sessió i sigui vàlid.

- Creem una *Is Not Null Unit* (Logged in?) per a comprovar que hi hagi un usuari a la sessió actual.
    - Unim l'*Input Collector* amb Logged in? amb un OK Link.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb Logged in? amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb l'*Input*.
- Creem una *Selector Unit* (Get User) de l'entitat Response per a extreure l'usuari de la base de dades.
    - Afegim una *Key Condition* a l'entitat User.
    - Unim Get user from session amb Get User amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb la *Key Condition*.
    - Unim Logged in? amb la unitat Get User amb un OK Link.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
    - Unim Get User amb Exists User amb un OK Link.
        - Coupling:
            - oidUser amb l'*Input*.
- Creem un *OK Collector* que anomenarem Modify User OK.
    - Afegim un *Output Collector Parameter* amb el nom email.
    - Afegim un *Output Collector Parameter* amb el nom oidUser.
    - Afegim un *Output Collector Parameter* amb el nom username.
    - Unim Get User amb Modify User OK amb un Transport Link.
        - Coupling:
            - email amb email.
            - oidUser amb oidUser.
            - username amb username.
    - Unim Exists User amb Modify User OK amb un OK Link.

En el cas d'aquest *Business Helper* tenim dues opcions d'error. Una és un error intern que no podem controlar i l'altra no hi hagi usuari a la sessió o no sigui correcte.

- Creem un *KO Collector* que anomenarem Internal error.
    - Unim Get User amb Internal error amb un KO Link.

- Creem un *KO Collector* que anomenarem Modify User KO.
    - Unim Logged in? amb Modify User KO amb un KO Link.
    - Unim Exists User amb Modify User KO amb un KO Link.



Il·lustració 57: Modify User BH (WebML)

7.3.13 Publish New Response BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Publish New Response BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- content
- oidThread

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Publish Response per a cridar la transacció TxCreateResponse per a crear una nova resposta.

- Creem una *Module Unit* (Publish Response) del tipus Publish Response.
    - Unim l'*Input Collector* amb Publish Response amb un OK Link.
        - Coupling:
            - content amb content.
            - oidThread amb oidThread.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb Publish Response amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb oidUser.

En el cas correcte volem retornar l'identificador del tema de discussió al que pertany la resposta per a seguir amb l'execució.

- Creem un *OK Collector* que anomenarem Publish Response OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Publish Response amb Publish Response OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim tres opcions d'error. Una és un error intern que no podem controlar, un altre és que el tema de discussió no sigui vàlid i l'altre és un error controlat en els paràmetres o en l'usuari i que volem redirigir amb el missatge d'error corresponent. Per tal de fer els missatges d'error tornarem a fer servir l'*Script Unit*.

- Creem un *KO Collector* que anomenarem Internal error.
    - Unim Publish Response amb Internal error amb un KO Link.
        - Collector Code: Internal Error.
- Creem un *KO Collector* que anomenarem Invalid thread.
    - Unim Publish Response amb Invalid thread amb un KO Link.
        - Collector Code: No Thread.

Les *Script Unit* les utiltizarem en el cas que l'usuari no existeixi o que el contingut sigui buit.

- Creem una *Script Unit* (Invalid user) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "You are not logged in. Please, try again later when logged in." ]
```

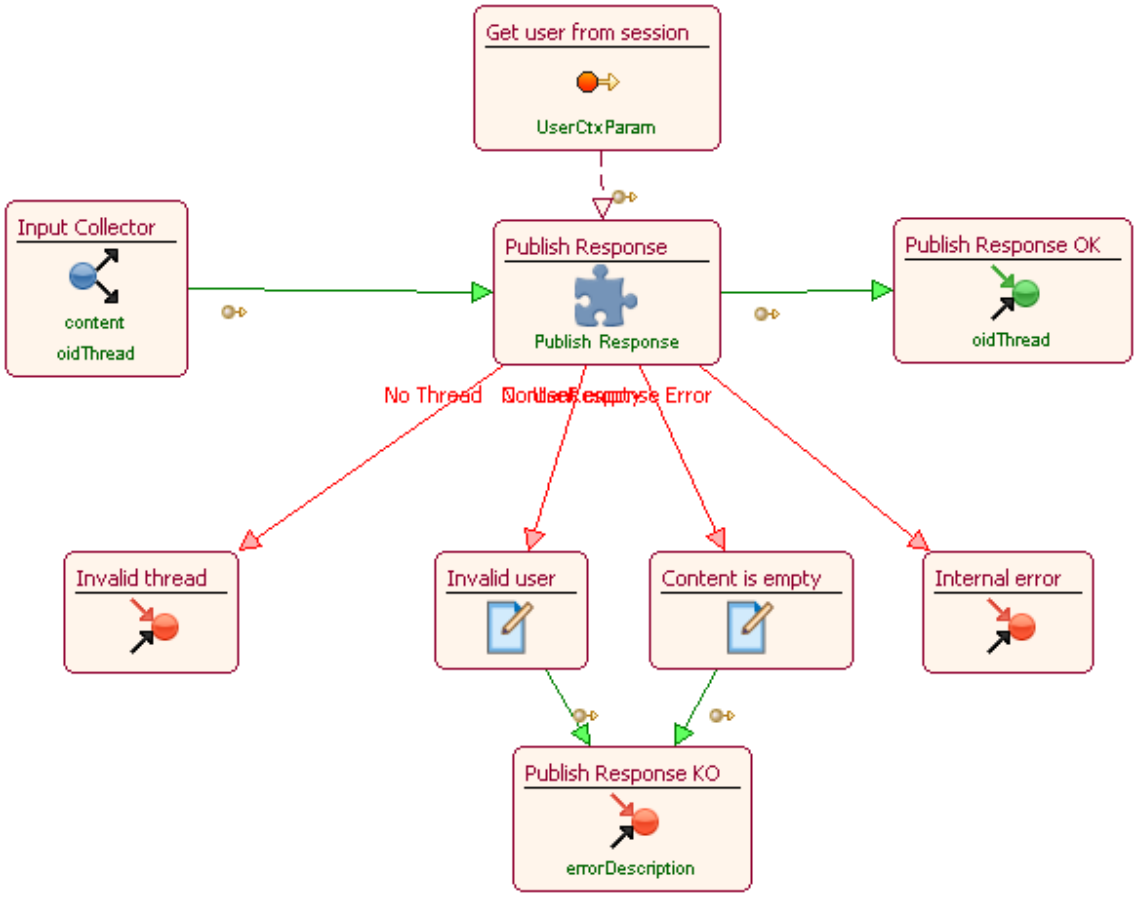
- Unim Publish Response amb Invalid user amb un KO Link.
  - Collector Code: No User.

- Creem una *Script Unit* (Content is empty) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Content in a response can not be empty." ]
```

- Unim Publish Response amb Content is empty amb un KO Link.
  - Collector Code: Content Empty.

- Creem un *KO Collector* que anomenarem Publish Response KO.
  - Afegim un *Output Collector Parameter* amb el nom errorDescription.
  - Unim Invalid user amb Publish Response KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Content is empty amb Publish Response KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.



Il·lustració 58: Publish New Response BH (WebML)

7.3.14 Publish New Thread BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Publish New Response BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- content
- content_type
- specialization
- title

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Publish Thread per a cridar la transacció TxCreateThread per a crear una nova resposta. Tots els paràmetres necessaris els obtenim d'entrada excepte l'usuari que s'extreu de la sessió.

- Creem una *Module Unit* (Publish Thread) del tipus Publish Thread.
    - Unim l'*Input Collector* amb Publish Thread amb un OK Link.
        - Coupling:
            - content amb content.
            - content_type amb content_type.
            - specialization amb specialization.
            - oidThread amb oidThread.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb Publish Thread amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb oidUser.

En cas de no tenir cap problema a l'hora de crear el tema de discussió, volem retornar el seu identificador.

- Creem un *OK Collector* que anomenarem Publish Thread OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Publish Thread amb Publish Thread OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim tres opcions d'error. Una és un error intern que no podem controlar, un altre és que l'usuari que intenta crear el tema de discussió no sigui vàlid i l'altre és un error controlat en els paràmetres i que volem redirigir amb el missatge d'error corresponent. Per tal de fer els missatges d'error tornarem a fer servir l'*Script Unit*.

- Creem un *KO Collector* que anomenarem Internal error.
    - Unim Publish Thread amb Internal error amb un KO Link.
        - Collector Code: Thread Error.
- Creem un *KO Collector* que anomenarem Invalid user.

- Unim Publish Thread amb Invalid user amb un KO Link.
  - Collector Code: No User.

Les *Script Unit* les utiltizarem en el cas que el tipus de contingut o l'especialitat no siguin vàlides o el títol o el contingut siguin buits.

- Creem una *Script Unit* (Invalid content type) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Invalid content type." ]
```

  - Unim Publish Response amb Invalid content type amb un KO Link.
    - Collector Code: No Content Type.
- Creem una *Script Unit* (Invalid specialization) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Invalid specialization." ]
```

  - Unim Publish Response amb Invalid specialization amb un KO Link.
    - Collector Code: No Specialization.
- Creem una *Script Unit* (Empty title) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Title can not be empty." ]
```

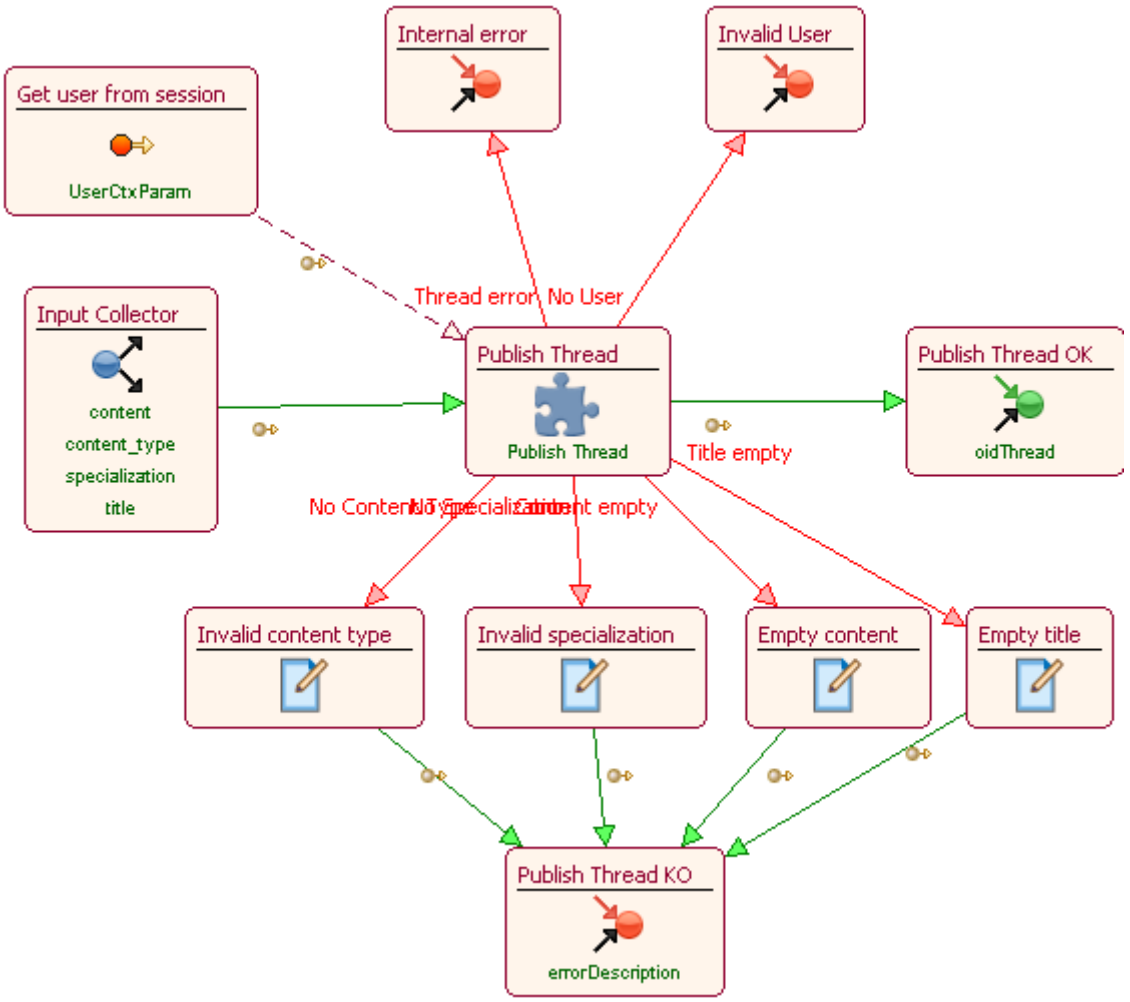
  - Unim Publish Response amb Empty title amb un KO Link.
    - Collector Code: Title empty.
- Creem una *Script Unit* (Empty content) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Content can not be empty." ]
```

  - Unim Publish Response amb Empty content amb un KO Link.
    - Collector Code: Content empty.

Unim totes les *Script Unit* al *KO Collector* corresponent.

- Creem un *KO Collector* que anomenarem Publish thread KO.
  - Afegim un *Output Collector Parameter* amb el nom errorDescription.
  - Unim Invalid content type amb Publish thread KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Invalid specialization amb Publish thread KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Empty title amb Publish thread KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Empty content amb Publish thread KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.



Il·lustració 59: Publish New Thread BH (WebML)

7.3.15 Register User BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Register User BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- email
- password
- specialization
- username

A continuació comencem a modelar el comportament del *business helper*. En aquest cas utilitzarem el mòdul Register per a cridar la transacció TxCreateUser per a crear una nou usuari al sistema. Tots els paràmetres necessaris els obtenim d'entrada. Un cop fet l'usuari el guardem automàticament en sessió.

- Creem una *Module Unit* (Register) del tipus Register.
  - Unim l'*Input Collector* amb Register amb un OK Link.
    - Coupling:
      - content amb content.
      - content_type amb content_type.
      - specialization amb specialization.
      - oidThread amb oidThread.
- Creem una *Set Unit* (Set user in session) amb UserCtxParam com a *Context Parameter*.
  - Unim Register amb Set user in session amb un Transport Link.
    - Collector Code: User Created.
    - Coupling:
      - oidUser amb UserCtxParam.oidUser.
- Creem un *OK Collector* que anomenarem Register OK.
  - Unim Register amb Register OK amb un OK Link.

Excepte en el cas d'error intern volem mostar un missatge referent a l'error a l'usuari. Per a fer-ho haurem de gestionar els diferents errors possibles del mòdul Register i generar els missatges a partir d'una *Script Unit*. Els possibles errors són que ja existeixi un usuari amb le mateix nom o e-mail, que l'especialitat no sigui vàlida o que algun dels paràmetres sigui buit.

- Creem un *KO Collector* que anomenarem Internal error.
  - Unim Register amb Internal error amb un KO Link.
    - Collector Code: User Error.
- Creem una *Script Unit* (Username exists) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "There's already an user with the same username. Try another one." ]
```

  - Unim Register amb Username exists amb un KO Link.
    - Collector Code: Username already exists.
- Creem una *Script Unit* (Email exists) amb el següent contingut:

```
#output String errorMsg
```

```
return [ "errorMsg" : "There's already an user with the same email. Try another one." ]
```

- Unim Register amb Email exists amb un KO Link.
  - Collector Code: Email already exists.

- Creem una *Script Unit* (Invalid specialization) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Invalid specialization." ]
```

- Unim Publish Response amb Invalid specialization amb un KO Link.
  - Collector Code: No Specialization.

- Creem una *Script Unit* (Empty username) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Username can not be empty." ]
```

- Unim Publish Response amb Empty username amb un KO Link.
  - Collector Code: Empty Title.

- Creem una *Script Unit* (Empty email) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Email can not be empty." ]
```

- Unim Publish Response amb Empty email amb un KO Link.
  - Collector Code: Empty Email.

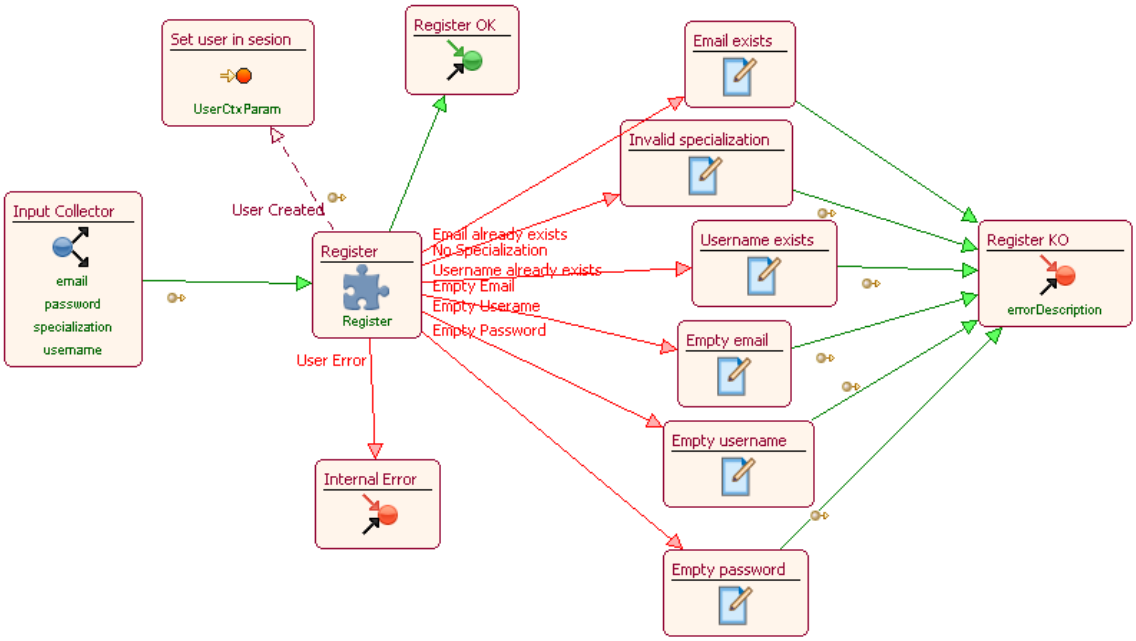
- Creem una *Script Unit* (Empty password) amb el següent contingut:

```
#output String errorMsg
return [ "errorMsg" : "Password can not be empty." ]
```

- Unim Publish Response amb Empty password amb un KO Link.
  - Collector Code: Empty Password.

Unim totes les *Script Unit* al *KO Collector* corresponent.

- Creem un *KO Collector* que anomenarem Register KO.
  - Afegim un *Output Collector Parameter* amb el nom errorDescription.
  - Unim Username exists amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Email exists amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Invalid specialization amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Empty username amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Empty email amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.
  - Unim Empty password amb Register KO amb un OK Link.
    - Coupling:
      - errorMsg amb errorDescription.



Il·lustració 60: Register User BH (WebML)

7.3.16 Remove Favorite BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Remove Favorite BH.

Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidThread

A continuació comencem a modelar el comportament del *business helper*. Principalment el que s'ha de fer és obtenir de la sessió l'usuari actual per a cridar la transacció TxRemoveFavorite i gestionar els resultats d'aquesta operació segons convingui.

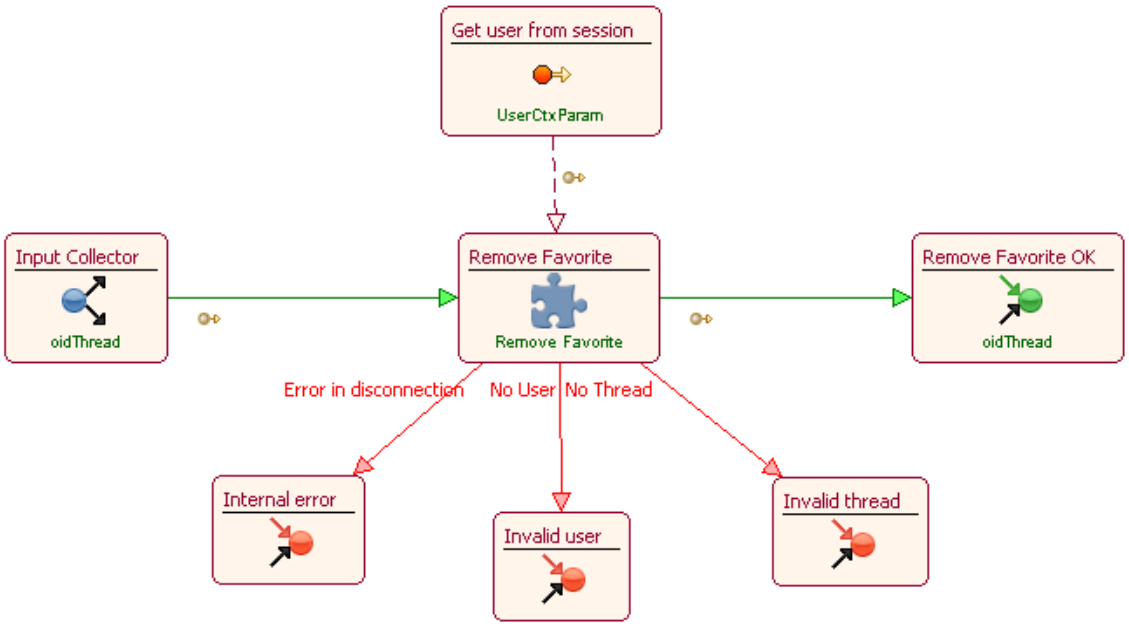
- Creem una *Module Unit* (Remove Favorite) del tipus Remove Favorite.
    - Unim l'*Input Collector* amb Remove Favorite amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
- Creem una *Get Unit* (Get user from session) amb UserCtxParam com a *Context Parameter*.
    - Unim Get user from session amb Remove Favorite amb un Transport Link.
        - Coupling:
            - UserCtxParam.oidUser amb oidUser.

En cas correcte volem retornar l'identificador del tema de discussió.

- Creem un *OK Collector* que anomenarem Remove Favorite OK.
    - Afegim un *Output Collector Parameter* amb el nom oidThread.
    - Unim Remove Favorite amb Remove Favorite OK amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.

En el cas d'aquest *Business Helper* tenim tres opcions d'error, una per a cada error del mòdul Remove Favorite.

- Creem un *KO Collector* que anomenarem Invalid User.
    - Unim Remove Favorite amb Invalid User.
        - Collector Code: No User.
- Creem un *KO Collector* que anomenarem Invalid Thread.
    - Unim Remove Favorite amb Invalid Thread.
        - Collector Code: No Thread.
- Creem un *KO Collector* que anomenarem Internal Error.
    - Unim Remove Favorite amb Internal Error.
        - Collector Code: Error in connection.



Il·lustració 61: Remove Favorite BH (WebML)

7.3.17 Thread Info BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem Thread Info BH.

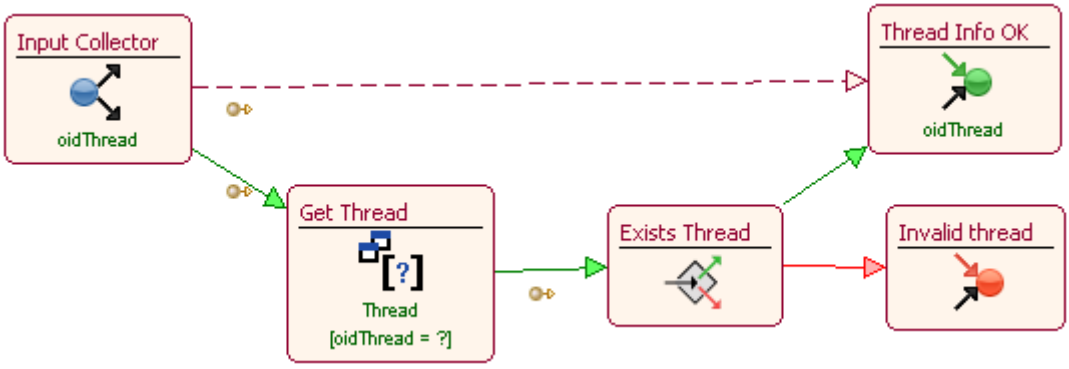
Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidThread

A continuació comencem a modelar el comportament del *business helper*. Tota la lògica que hi havia en aquest *business helper* relacionada amb l'extracció de la informació del tema de discussió desapareix perquè s'incorpora a la *Server Page*. De totes formes, ens quedaria fer la comprovació de que existeix el tema per a poder redirigir en el cas necessari ja que això no ho podem fer des de les pàgines.

Com ja em fet altres vegades, extreurem de la base de dades el tema de discussió pertanyent a l'identificador però no rebrem cap error per part de la base de dades en el cas de que no existeixi sinó un resultat buit. Això ho tractarem amb una *Is Not Null Unit*.

- Creem una *Selector Unit* (Get Thread) de l'entitat Thread per a extreure el tema de discussió de la base de dades.
  - Afegim una *Key Condition* a l'entitat Thread.
  - Unim l'*Input Collector* amb Get Thread amb un OK Link.
    - Coupling:
      - oidThread amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists Thread).
  - Unim Get Thread amb Exists Thread amb un OK Link.
    - Coupling:
      - oidThread amb l'*Input*.
  - Unim Exists User amb un *KO Collector* que anomenarem Invalid thread amb un KO Link.
- Creem un *OK Collector* que anomenarem Thread Info OK.
  - Afegim un *Output Collector Parameter* amb el nom oidThread.
  - Unim l'*Input Collector* amb Thread Info OK amb un Transport Link.
    - Coupling:
      - oidThread amb oidThread.



Il·lustració 62: Thread Info BH (WebML)

7.3.18 User Profile BH

Comencem creant un nou mòdul dins dels Forum Business Helpers. En aquest cas l'anomenarem User Profile BH.

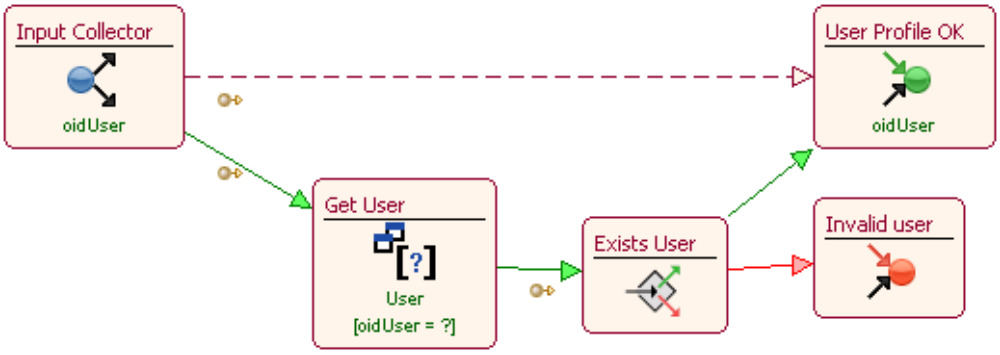
Dins del mòdul crearem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada variable que necessita el *business helper*.

- oidUser

A continuació comencem a modelar el comportament del *business helper*. Tota la lògica que hi havia en aquest *business helper* relacionada amb l'extracció de la informació de l'usuari desapareix perquè s'incorpora a la *Server Page*. De totes formes, ens quedaria fer la comprovació de que existeix l'usuari per a poder redirigir en el cas necessari ja que això no ho podem fer des de les pàgines.

Com ja em fet altres vegades, extreurem de la base de dades l'usuari pertanyent a l'identificador però no rebrem cap error per part de la base de dades en el cas de que no existeixi sinó un resultat buit. Això ho tractarem amb una *Is Not Null Unit*.

- Creem una *Selector Unit* (Get User) de l'entitat User per a extreure el tema de discussió de la base de dades.
  - Afegim una *Key Condition* a l'entitat User.
  - Unim l'*Input Collector* amb Get User amb un OK Link.
    - Coupling:
      - oidUser amb la *Key Condition*.
- Per tal de tractar el resultat, afegim una *Is Not Null Unit* (Exists User).
  - Unim Get User amb Exists User amb un OK Link.
    - Coupling:
      - oidUser amb l'*Input*.
  - Unim Exists User amb un *KO Collector* que anomenarem Invalid user amb un KO Link.
- Creem un *OK Collector* que anomenarem User Profile OK.
  - Afegim un *Output Collector Parameter* amb el nom oidUser.
  - Unim l'*Input Collector* amb User Profile OK amb un Transport Link.
    - Coupling:
      - oidUser amb oidUser.



Il·lustració 63: User Profile BH (WebML)

7.4 Pàgines

En aquest apartat s'explica com s'han generat les pàgines de l'aplicació. Aquestes corresponen bàsicament a les *server pages* que construïen la pàgina que rep l'usuari però amb alguna lògica d'extracció de la informació que hem eliminat dels *business helpers*.

Les unitats de contingut, com es va explicar al principi, permeten l'ús de selectors per a escollir les instàncies que es mostraran d'una entitat. Com aquestes unitats són les mateixes que s'usen per a mostrar informació a les pàgines, les hem usat junt amb els selectors per a fer la tria des d'allà.

Així doncs, es podrà veure quin contingut té cada pàgina a partir de la informació de l'UX Model i com aquesta s'ha traduït a WebML incorporant certa lògica dels *business helper*.

Cal destacar que els links d'unió amb altres pàgines o *business helper* no estan aquí explicats ja que es veurà en a l'apartat "Únio de les diferents capes".

7.4.1 Create Response

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Create Response.

Tal i com es veu a l'UX Model, la pàgina de creació de resposta està composta bàsicament per un formulari de creació de resposta i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

L'atribut *Hidden*, permet no mostrar el camp i ens servirà en el cas de l'identificador del tema de discussió per a saber a quin tema pertany la resposta sense que l'usuari ho vegi.

- Creem una *Entry Unit* (Response form) per a introduir les dades per a la creació de la resposta.
    - Afegim un camp de tipus *Field* amb nom "Content" que tindrà tipus Text i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "oidThread" que tindrà tipus Integer.
        - Preloaded, Hidden.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

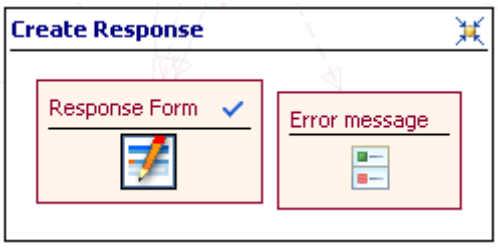
Això ens permet obligar a estar omplert el camp del contingut abans d'enviar el formulari.

- Afegim al camp amb nom "content" de Response Form una *Validation Rule* del tipus Mandatory.

Podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut:
  $$errorMsg$$



Il·lustració 64: Create Response Page (WebML)

7.4.2 Create Thread

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Create Thread.

Tal i com es veu a l'UX Model, la pàgina de creació de tema de discussió està composta bàsicament per un formulari de creació de tema de discussió i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

- Creem una *Entry Unit* (Response form) per a introduir les dades per a la creació de la resposta.
    - Afegim un camp de tipus *Field* amb nom "Title" que tindrà tipus String i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Content" que tindrà tipus Text i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Selection Field* amb nom "Content Type" que tindrà tipus String i text/plain.
        - Modifiable.
        - *Slot* amb *Label* i *Output*.
    - Afegim un camp de tipus *Selection Field* amb nom "Specialization" que tindrà tipus String i text/plain.
        - Modifiable.
        - *Slot* amb *Label* i *Output*.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

Això ens permet obligar a estar omplert el camp del contingut abans d'enviar el formulari.

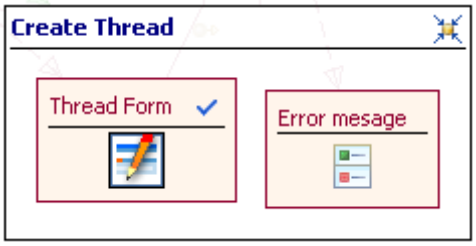
- Afegim al camp amb nom "Title" de Response Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Content" de Response Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Content Type" de Response Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Specialization" de Response Form una *Validation Rule* del tipus Mandatory.

Podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

Un cop fet el formulari, podem reordenar els camps com més ens convingui seleccionant l'*Entry Unit* dins la pàgina i canviar el *Field Order*.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut: $$errorMsg$$



Il·lustració 65: Create Thread Page (WebML)

7.4.3 Create User

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Create User.

Tal i com es veu a l'UX Model, la pàgina de creació d'usuari està composta bàsicament per un formulari de creació d'usuari i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

- Creem una *Entry Unit* (User Form) per a introduir les dades per a la creació de l'usuari.
    - Afegim un camp de tipus *Field* amb nom "Username" que tindrà tipus String i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Email" que tindrà tipus String i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Password" que tindrà tipus Password i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Selection Field* amb nom "Specialization" que tindrà tipus String i text/plain.
        - Modifiable.
        - *Slot* amb *Label* i *Output*.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

En el cas de la creació de l'usuari, tots els camps hauríem d'estar omplerts i en el cas l'email tenir l'estructura correcta.

- Afegim al camp amb nom "Username" de User Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Email" de User Form una *Validation Rule* del tipus EMail.
- Afegim al camp amb nom "Password" de User Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Specialization" de User Form una *Validation Rule* del tipus Mandatory.

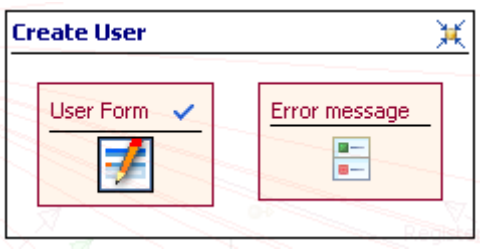
En tots els casos, podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

Un cop fet el formulari, podem reordenar els camps com més ens convingui seleccionant l'*Entry Unit* dins la pàgina i canviar el *Field Order*.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure

qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut: $$errorMsg$$



Il·lustració 66: Create User Page (WebML)

7.4.4 Index

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Index.

Tal i com es veu a l'UX Model, la pàgina d'índex està composta d'un formulari que farà de filtre, la informació simplificada dels temes de discussió que passin el filtre i el llistat de temes de discussió preferits de l'usuari a sessió. Les dues llistes tindran la seva corresponent paginació amb un màxim de 10 per pàgina.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

- Creem una *Entry Unit* (Index Selector) per a introduir les dades per a la creació de l'usuari.
    - Afegim un camp de tipus *Selection Field* amb nom "Content Type" que tindrà tipus String i text/plain.
        - Modifiable.
        - *Slot* amb *Label* i *Output*.
    - Afegim un camp de tipus *Selection Field* amb nom "Specialization" que tindrà tipus String i text/plain.
        - Modifiable.
        - *Slot* amb *Label* i *Output*.

Les *Powered Index Units* són un tipus de *Content Unit* que ens permet llistar informació d'instàncies d'una entitat donats uns filtres i que permet generar una paginació automàticament. Aquest tipus d'unitat la farem servir per a les llistes de temes de discussió filtrats i temes de discussió preferits en aquesta pàgina.

- Creem una *Power Index Unit* (Matching) de l'entitat Thread.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - title
            - date
            - HasResponses.oidResponse (count)
            - HasType.type
            - HasSpecialization.name
            - PostedBy.userName
    - Ordenem les instàncies per la data de creació en sentit descendent.
        - Sort Attributes: date[desc].
    - Per tal de fer la paginació, modificarem el *Block Factor*.
        - Block Factor: 10
- Creem una *Power Index Unit* (Favorites) de l'entitat Thread.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - title
            - date
            - HasResponses.oidResponse (count)

        - HasType.type
        - HasSpecialization.name
        - PostedBy.userName
    - Ordenem les instàncies per la data de creació en sentit descendent.
        - Sort Attributes: date[desc].
    - Per tal de fer la paginació, modificarem el *Block Factor*.
        - Block Factor: 10

Ara per ara es mostren tots els temes de discussió dues vegades ja que no tenim cap tipus de filtre. En el cas de Matching volem filtrar-los segons el seu tipus de contingut i especialitat. Com això funciona a partir del formulari de la mateixa pàgina, podem afegir ja un link per a fer l'emparellament de paràmetres un cop afegim els filtres necessaris.

- Modifiquem Matching per a filtrar els temes de discussió.
    - Afegim una *Attribute Condition* per a l'atribut "HasType.type" de l'entitat Thread.
    - Afegim una *Attribute Condition* per a l'atribut "HasSpecialization.name" de l'entitat Thread.

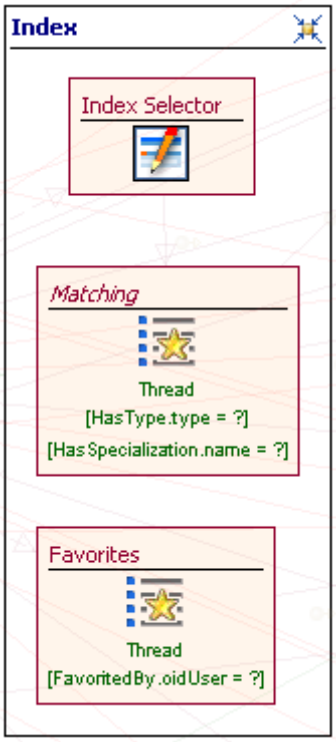
El filtratge per defecte, si no s'indica el contrari en algun dels casos, és mostrar tots els temes de discussió. Per això, haurem de forçar que en cas de ser *null* un dels paràmetres la condició sigui *true* per a aquell paràmetre. Notar que content_type i specialization són els noms que hem donat a les *Attribute Condition*.

- Modifiquem el *Selector* de Matching editant l'*Expression*.
    - Afegim una nova condició.
        - If null true.
        - content_type.
    - Afegim una nova condició.
        - AND.
        - If null true.
        - specialization.
- Un cop ja tenim la condició creada, enllacem el formulari amb la *Power Index Unit*.
    - Unim Index Selector amb Matching amb un Normal Link.
        - Name: Change preferences.
        - Coupling:
            - Content Type amb content_type.
            - Specialization amb specialization.

Per acabar afegirem la condició a Favorites per a que més tard es puguin filtrar els missatges quan tinguem la informació necessària.

- Modifiquem Favorites per a filtrar els temes de discussió.
    - Afegim una *Attribute Condition* per a l'atribut FavoritedBy.oidUser de l'entitat Thread.

S'ha de vigilar de deshabilitar el *count* que acostuma a sortir quan hi ha més d'una entitat relacionada.



Il·lustració 67: Index Page (WebML)

7.4.5 Internal Error

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Internal Error.

Aquesta pàgina s'ha afegit a WebRatio per tal de controlar els errors interns de l'aplicació. Vindria a ser la pàgina per defecte en un cas d'error 500.

L'únic que volem mostrar en aquests casos és un missatge d'error demanant disculpes a l'usuari per a aquest problema.

- Creem una *Multi Message Unit* (Internal error message) amb el següent contingut:
  The site has experimented some kind of internal error.
  Our apologizes for these problem.
  Try again later in order to check if it has been solved.
  Thanks for your patience!

Il·lustració 68: Internal Error Page (WebML)

7.4.6 Login

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Login.

Tal i com es veu a l'UX Model, la pàgina de login està composta bàsicament per un formulari de d'entrada i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

- Creem una *Entry Unit* (Login Form) per a introduir les dades per a la creació de l'usuari.
    - Afegim un camp de tipus *Field* amb nom "Username" que tindrà tipus String i text/plain.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Password" que tindrà tipus Password i text/plain.
        - Modifiable.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

En el cas del login, tots els camps hauríem d'estar omplerts.

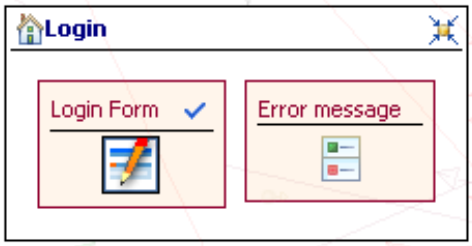
- Afegim al camp amb nom "Username" de Login Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Password" de Login Form una *Validation Rule* del tipus Mandatory.

Un cop fet el formulari, podem reordenar els camps com més ens convingui seleccionant l'*Entry Unit* dins la pàgina i canviar el *Field Order*.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut:
  $$errorMsg$$

Com la pàgina de login serà la pàgina inicial de l'aplicació, la marquem com a *Home* a la pestanya de *Properties* de la pàgina.



Il·lustració 69: Login Page (WebML)

7.4.7 Modify Response

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Modify Response.

Tal i com es veu a l'UX Model, la pàgina de modificació de resposta està composta bàsicament per un formulari de modificació de resposta i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

L'atribut *Hidden*, permet no mostrar el camp i ens servirà en el cas de l'identificador de la resposta per a saber quina és la resposta que s'ha de modificar sense que l'usuari ho vegi.

A més a més és necessari afegir l'atribut *Preloaded* al contingut per a poder mostrar el contingut actual a l'usuari abans de la modificació.

- Creem una *Entry Unit* (Modify Response form) per a introduir les dades per a la creació de la resposta.
    - Afegim un camp de tipus *Field* amb nom "Content" que tindrà tipus Text i text/plain.
        - Preloaded, Modifiable.
    - Afegim un camp de tipus *Field* amb nom "oidResponse" que tindrà tipus Integer.
        - Preloaded, Hidden.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

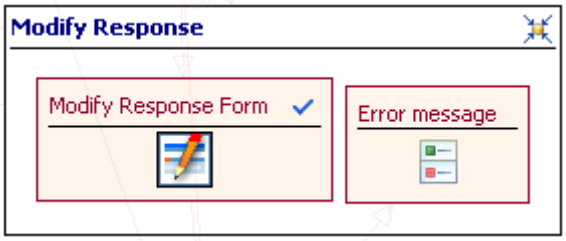
Això ens permet obligar a estar omplert el camp del contingut abans d'enviar el formulari.

- Afegim al camp amb nom "Content" de Modify Response Form una *Validation Rule* del tipus Mandatory.

Podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut:
  $$errorMsg$$



Il·lustració 70: Modify Response Page (WebML)

7.4.8 Modify Thread

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Modify Thread.

Tal i com es veu a l'UX Model, la pàgina de modificació de tema de discussió està composta bàsicament per un formulari de modificació de tema de discussió i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

L'atribut *Hidden*, permet no mostrar el camp i ens servirà en el cas de l'identificador del tema de discussió per a saber quina és el tema que s'ha de modificar sense que l'usuari ho vegi.

A més a més és necessari afegir l'atribut *Preloaded* a la resta de camps per a poder mostrar el seu contingut actual a l'usuari abans de la modificació.

- Creem una *Entry Unit* (Modify Thread Form) per a introduir les dades per a la creació de la resposta.
    - Afegim un camp de tipus *Field* amb nom "Title" que tindrà tipus String i text/plain.
        - Preloaded, Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Content" que tindrà tipus Text i text/plain.
        - Preloaded, Modifiable.
    - Afegim un camp de tipus *Selection Field* amb nom "Content Type" que tindrà tipus String i text/plain.
        - Preloaded, Modifiable.
        - *Slot* amb *Label* i *Output*.
    - Afegim un camp de tipus *Selection Field* amb nom "Specialization" que tindrà tipus String i text/plain.
        - Preloaded, Modifiable.
        - *Slot* amb *Label* i *Output*.
    - Afegim un camp de tipus *Field* amb nom "oidResponse" que tindrà tipus Integer.
        - Preloaded, Hidden.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

Això ens permet obligar a estar omplert el camp del contingut abans d'enviar el formulari.

- Afegim al camp amb nom "Title" de Response Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Content" de Response Form una *Validation Rule* del tipus Mandatory.
- Afegim al camp amb nom "Content Type" de Response Form una *Validation Rule* del tipus Mandatory.

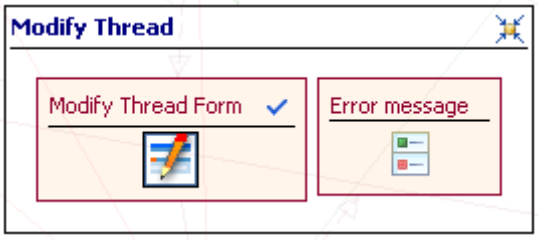
- Afegim al camp amb nom "Specialization" de Response Form una *Validation Rule* del tipus Mandatory.

Podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

Un cop fet el formulari, podem reordenar els camps com més ens convingui seleccionant l'*Entry Unit* dins la pàgina i canviar el *Field Order*.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut:
  $$errorMsg$$



Il·lustració 71: Modify Thread Page (WebML)

7.4.9 Modify User

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Modify User.

Tal i com es veu a l'UX Model, la pàgina de modificació d'usuari està composta bàsicament per un formulari de modificació d'usuari i una secció d'errors.

Per al formulari, només caldrà afegir-lo dins la pàgina i dotar-lo dels camps corresponents. Els noms que donem als camps són els noms amb els que es mostraran al formar la pàgina.

L'atribut *Hidden*, permet no mostrar el camp i ens servirà en el cas de l'identificador del tema de discussió per a saber quina és el tema que s'ha de modificar sense que l'usuari ho vegi.

A més a més és necessari afegir l'atribut *Preloaded* a la resta de camps per a poder mostrar el seu contingut actual a l'usuari abans de la modificació.

En el cas del nom d'usuari, el volem mostrar però que no es pugui modificar. Per això eliminem l'opció *Modifiable* del camp corresponent.

- Creem una *Entry Unit* (Modify User Form) per a introduir les dades per a la creació de la resposta.
    - Afegim un camp de tipus *Field* amb nom "Username" que tindrà tipus String i text/plain.
        - Preloaded.
    - Afegim un camp de tipus *Field* amb nom "Email" que tindrà tipus String i text/plain.
        - Preloaded, Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Current Password" que tindrà tipus Password.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "New password" que tindrà tipus Password.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "Repeat new password" que tindrà tipus Password.
        - Modifiable.
    - Afegim un camp de tipus *Field* amb nom "oidUser" que tindrà tipus Integer.
        - Preloaded, Hidden.

Igual que podríem fer amb Javascript, WebRatio ofereix la possibilitat d'afegir validacions als camps abans d'enviar el formulari. Aquesta opció ens serveix per a obligar a omplir certs camps en aquest cas. Tot i així, serà necessari tornar-ho a comprovar des de la capa de domini tal i com fèiem abans, això només és una comprovació extra per a ajudar a l'usuari.

Gràcies a això podem filtrar des de la capa de presentació que els dos nous e-mails siguin iguals. De fet, aquesta restricció és simplement per a l'usuari ja que l'aplicació només farà cas de la primera entrada.

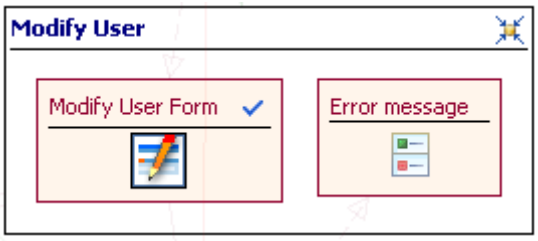
- Afegim al camp amb nom "Repeate new password" de Response Form una *Validation Rule* del tipus Compare.
    - Modifiquem *Value Field* per a que sigui New password.

Podem donar el nom que vulguem a les *Validation Rules* i el missatge d'error que volem que es mostri quan no es compleixi.

Un cop fet el formulari, podem reordenar els camps com més ens convingui seleccionant l'*Entry Unit* dins la pàgina i canviar el *Field Order*.

En el cas del missatge d'error, serà necessari crear una *Multi Message Unit* per a poder mostrar el missatge d'error que faci falta. En una *Multi Message Unit* podem escriure qualsevol text amb un *placeholder* per a les variables. Els noms dels *placeholder* ens serviran després per a poder fer la unió del missatge i la unitat i es defineixen usant dos símbols $ davant i darrera del nom.

- Creem una *Multi Message Unit* (Error message) amb el següent contingut: $$errorMsg$$



Il·lustració 72: Modify User Page (WebML)

7.4.10 Thread Info

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem Thread Info.

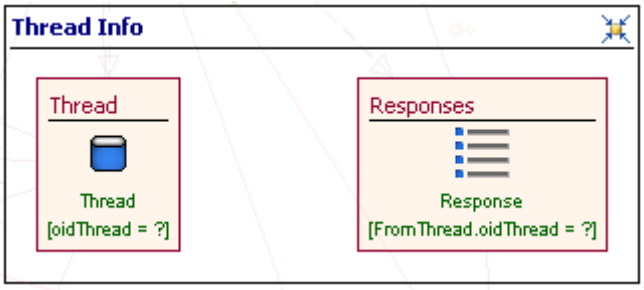
Tal i com es veu a l'UX Model, la pàgina d'informació d'un tema de discussió està composta bàsicament per la informació del tema i la de les seves respostes.

Per a mostrar la informació del tema de discussió s'usarà una *Data Unit*. Aquesta unitat permet mostrar la informació d'una sola instància a partir de la seva clau.

- Creem una *Data Unit* (Thread) de l'entitat Thread per a mostrar la informació el tema de discussió.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - title
            - content
            - date
            - PostedBy.userName
            - HasType.type
            - HasSpecialization.name
            - modified_date

En el cas de les respostes volem mostrar la informació de totes les que siguin del tema de discussió. Per a això s'usarà una *Index Unit* que permet mostrar una seria d'instàncies d'una entitat a partir de condicions fixades.

- Creem una *Index Unit* de l'entitat Response per a mostrar les respostes.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - content
            - date
            - PostedBy.userName
            - FromUser.HasSpecialization.specialization
            - HasSpecialization.name
            - modified_date
    - Afegim una *Attribute Condition* per a l'atribut "FromThread.oidThread" de l'entitat Response.



Il·lustració 73: Thread Info Page (WebML)

7.4.11 User Profile

Comencem creant una nova pàgina dins de la *Site View* de l'aplicació. En aquest cas l'anomenarem User Profile.

Tal i com es veu a l'UX Model, la pàgina d'informació d'un tema de discussió està composta bàsicament per la informació del tema i la de les seves respostes.

Per a mostrar la informació del tema de discussió s'usarà una *Data Unit*. Aquesta unitat permet mostrar la informació d'una sola instància a partir de la seva clau.

- Creem una *Data Unit* (Thread) de l'entitat Thread per a mostrar la informació el tema de discussió.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - userName
            - email
            - HasSpecialization.name
            - creation_time
            - last_connection

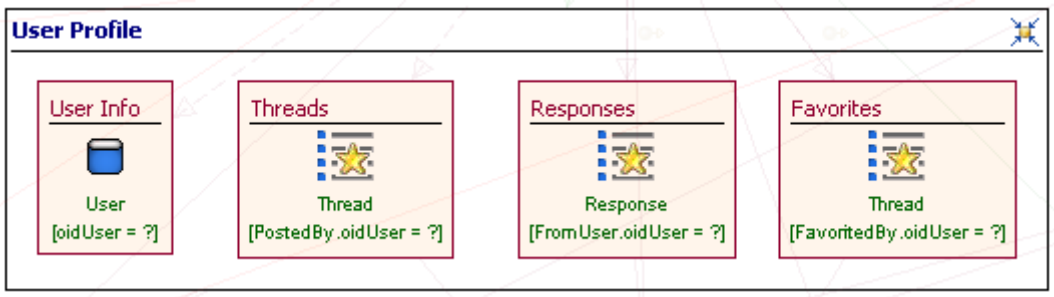
Les *Powered Index Units* són un tipus de *Content Unit* que ens permet llistar informació d'instàncies d'una entitat donats uns filtres i que permet generar una paginació automàticament. Aquest tipus d'unitat la farem servir per a les llistes de temes de discussió de l'usuari, les respostes de l'usuari i els temes de discussió preferits per l'usuari a la pàgina.

- Creem una *Power Index Unit* (Threads) de l'entitat Thread.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - title
            - date
            - HasResponses.oidResponse (count)
            - HasType.type
            - HasSpecialization.name
    - Ordenem les instàncies per la data de creació en sentit descendent.
        - Sort Attributes: date[desc].
    - Per tal de fer la paginació, modificarem el *Block Factor*.
        - Block Factor: 10
- Creem una *Power Index Unit* (Responses) de l'entitat Thread.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - content
            - date
            - FromThread.title
            - FromThread.date
            - FromThread.PostedBy.userName
            - FromThread.HasType.type

        - FromThread.HasSpecialization.name
    - Ordenem les instàncies per la data de creació en sentit descendent.
        - Sort Attributes: date[desc].
    - Per tal de fer la paginació, modificarem el *Block Factor*.
        - Block Factor: 10
- Creem una *Power Index Unit* (Favorites) de l'entitat Thread.
    - Escollirem a mostrar els paràmetres segons l'UX Model.
        - Display Attributes:
            - title
            - date
            - HasResponses.oidResponse (count)
            - HasType.type
            - HasSpecialization.name
            - PostedBy.userName
    - Per tal de fer la paginació, modificarem el *Block Factor*.
        - Block Factor: 10

Ara per ara les diferents *Power Index Unit* no tenen cap tipus de condició per a mostrar o no certa entitat. Per això haurem d'afegir *Attribute Conditions* per a després filtrar les corresponents a l'usuari que es mostra.

- Modifiquem Threads per a filtrar els temes de discussió.
    - Afegim una *Attribute Condition* per a l'atribut "PostedBy.oidUser" de l'entitat Thread.
- Modifiquem Responses per a filtrar les respostes.
    - Afegim una *Attribute Condition* per a l'atribut "FromUser.oidUser" de l'entitat Response.
- Modifiquem Favorites per a filtrar els temes de discussió preferits de l'usuari.
    - Afegim una *Attribute Condition* per a l'atribut "FavoritedBy.oidUser" de l'entitat Thread.



Il·lustració 74: User Profile Page (WebML)

7.5 Unió de les diferents capes

A partir del diagrama intern que hem definit a l'UML hauríem de ser capaços d'enllaçar les pàgines ara creades amb els *business helper* que controlen les transaccions i les condicions de la informació a mostrar per les pàgines.

A continuació el que es mostren són els passos a seguir per a modificar la *Site View* i crear tot els links necessaris per a unir l'aplicació.

7.5.1 Business Helpers

Crearem una *Module Unit* a la *site view* per a cada *business helper* que necessitem i l'enllaçarem amb les pàgines necessàries.

En el cas de nombrar alguna unitat que encara no existeix, es pot crear sense cap link. Sempre es segueix el patró de nombrar les *Module Unit* amb el mateix nom dels *business helper*.

7.5.1.1 Add Favoirte BH

- Creem una *Module Unit* (Add Favorite BH) del mòdul Add Favorite BH.
    - Unim Add Favorite BH amb Thread Info BH amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Add Favorite BH amb Index BH amb un KO Link.
        - Collector Code: Invalid thread.
    - Unim Add Favorite BH amb Login amb un KO Link.
        - Collector Code: Invalid user.
    - Unim Add Favorite BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.2 Create Thread BH

- Creem una *Module Unit* (Create Thread BH) del mòdul Create Thread BH.
    - Unim Create Thread BH amb Create Thread amb un OK Link.
    - Unim Create Thread BH amb Thread Form de Create Thread amb un Transport Link.
        - Collector Code: Create Thread OK.
        - Coupling:
            - contentTypes amb la llista de Content Type.
            - specializations amb la llista d'Specialization.
    - Unim Create Thread BH amb Internal Error amb un KO Link.

7.5.1.3 Create User BH

- Creem una *Module Unit* (Create User BH) del mòdul Create User BH.
    - Unim Create User BH amb Create User amb un OK Link.
    - Unim Create User BH amb User Form de Create User amb un Transport Link.
        - Collector Code: Create user OK.
        - Coupling:
            - specializations amb Specialization.
    - Unim Create User BH amb Internal Error amb un KO Link.

7.5.1.4 Do Login BH

- Creem una *Module Unit* (Do Login BH) del mòdul Do Login BH.
    - Unim Do Login BH amb Index BH amb un OK Link.
    - Unim Do Login BH amb Error message de Login amb un Transport Link.
        - Collector Code: Do Login KO.
        - Coupling:
            - errorDescription amb errorMsg.
    - Unim Do Login BH amb Login amb un KO Link.
        - Collector Code: Login KO.
    - Unim Do Login BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.5 Do Logout BH

- Creem una *Modeule Unit* (Do Logout BH) del mòdul Do Logout BH.
    - Unim Logout BH amb Login amb un OK Link.

7.5.1.6 Do Response Modification BH

- Creem una *Module Unit* (Do Response Modification BH) del mòdul Do Response Modification BH.
    - Unim Do Response Modification BH amb Thread Info BH amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Do Response Modification BH amb Error message de Modify Response amb un Transport Link.
        - Collector Code: Response Modification KO.
        - Coupling:
            - errorDescription amb errorMsg.
    - Unim Do Response Modification BH amb Modify Response BH amb un KO Link.
        - Collector Code: Response Modification KO.
        - Coupling:
            - oidResponse amb oidResponse.
    - Unim Do Response Modification BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.7 Do Thread Modification BH

- Creem una *Module Unit* (Do Thread Modification BH) del mòdul Do Thread Modification BH.
    - Unim Do Thread Modification BH amb Thread Info BH amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Do Thread Modification BH amb Error message de Modify Response amb un Transport Link.
        - Collector Code: Response Modification KO.
        - Coupling:
            - errorDescription amb errorMsg.

- Unim Do Thread Modification BH amb Modify Thread BH amb un KO Link.
  - Collector Code: Thread Modification KO.
  - Coupling:
    - oidThread amb oidThread.
- Unim Do Thread Modification BH amb Index BH amb un KO Link.
  - Collector Code: Invalid thread.
- Unim Do Thread Modification BH amb Internal Error amb un KO Link.
  - Collector Code: Internal error.

7.5.1.8 Do User Modification BH

- Creem una *Module Unit* (Do User Modification BH) del mòdul Do User Modification BH.
  - Unim Do User Modification BH amb User Profile BH amb un OK Link.
    - Coupling:
      - oidUser amb oidUser.
  - Unim Do User Modification BH amb Error message de Modify User amb un Transport Link.
    - Collector Code: User Modification KO.
    - Coupling:
      - errorDescription amb errorMsg.
  - Unim Do User Modification BH amb Modify User BH amb un KO Link.
    - Collector Code: User Modification KO.
  - Unim Do User Modification BH amb Internal Error amb un KO Link.
    - Collector Code: Internal error.

7.5.1.9 Index BH

- Creem una *Module Unit* (Index BH) del mòdul Index BH.
  - Unim Index BH amb Index amb un OK Link.
  - Unim Index BH amb Index Selector d'Index amb un Transport Link.
    - Collector Code: Index OK.
    - Coupling:
      - contentTypes amb Content Type.
      - specializations amb Specialization.
  - Unim Index BH amb Internal Error amb un KO Link.
- Modifiquem Favorites d'Index per a filtrar segons l'usuari.
  - Afegim una *Attribute Condition* per a l'atribut "FavoritedBy.oidUser" de l'entitat Thread.
  - Unim Index BH amb Favorites d'Index amb un Transport Link.
    - Coupling:
      - oidUser amb la *Key Condition*.

7.5.1.10 Modify Response BH

- Creem una *Module Unit* (Modify Response BH) del mòdul Modify Response BH.
  - Unim Modify Response BH amb Modify Response amb un OK Link.
  - Unim Modify Response BH amb Modify Response Form amb un Transport Link.

        - Collector Code: Modify Response OK.
        - Coupling:
            - content amb Content.
            - oidResponse amb oidResponse
    - Unim Modify Response BH amb Index BH amb un KO Link.
        - Collector Code: Invalid response.
    - Unim Modify Response BH amb Thread Info BH amb un KO Link.
        - Collector Code: Invalid user.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Modify Response BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.11 Modify Thread BH

- Creem una *Module Unit* (Modify Thread BH) del mòdul Modify Thread BH.
    - Unim Modify Thread BH amb Modify Thread amb un OK Link.
    - Unim Modify Thread BH amb Modify Thread Form amb un Transport Link.
        - Collector Code: Modify Thread OK.
        - Coupling:
            - content amb Content.
            - content_type_list amb els valors de Content Type.
            - content_type amb la preselecció de Content Type.
            - oidThread amb oidThread.
            - specialization_list amb els valors d'Specialization.
            - specialization amb la preselecció d'Specialization.
            - title amb title.
    - Unim Modify Thread BH amb Index BH amb un KO Link.
        - Collector Code: Invalid thread.
    - Unim Modify Thread BH amb Thread Info BH amb un KO Link.
        - Collector Code: Modify Thread KO.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Modify Thread BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.12 Modify User BH

- Creem una *Module Unit* (Modify User BH) del mòdul Modify User BH.
    - Unim Modify User BH amb Modify User amb un OK Link.
    - Unim Modify User BH amb Modify User Form amb un Transport Link.
        - Collector Code: Modify User OK.
        - Coupling:
            - email amb Email.
            - oidUser amb oidUser.
            - username amb Username.

- Unim Modify User BH amb Login amb un KO Link.
  - Collector Code: Modify User KO.
- Unim Modify User BH amb Internal Error amb un KO Link.
  - Collector Code: Internal error.

7.5.1.13 Publish New Response BH

- Creem una *Module Unit* (Publish New Response BH) del mòdul Publish New Response BH.
  - Unim Publish New Response BH amb Error message de Create Response amb un Transport Link.
    - Collector Code: Publish Response KO.
    - Coupling:
      - errorDescription amb errorMsg.
  - Unim Publish New Response BH amb Create Response amb un KO Link.
    - Collector Code: Create Response KO.
  - Unim Publish New Response BH amb Index BH amb un KO Link.
    - Collector Code: Invlaid thread.
  - Unim Publish New Response BH amb Internal Error amb un KO Link.
    - Collector Code: Internal error.
  - Unim Publish New Response BH amb Thread Info BH un OK Link.
    - Coupling:
      - oidThread amb oidThread.

7.5.1.14 Publish New Thread BH

- Creem una *Module Unit* (Publish New Thread BH) del mòdul Publish New Thread BH.
  - Unim Publish New Thread BH amb Thread Info BH amb un OK Link.
    - Coupling:
      - oidThread amb oidThread.
  - Unim Publish New Thread BH amb Error message de Create Thread amb un Transport Link.
    - Coupling:
      - errorDescription amb errorMsg.
  - Unim Publish New Thread BH amb Create Thread BH amb un KO Link.
    - Collector Code: Publish Thread KO.
  - Unim Publish New Thread BH amb Login amb un KO Link.
    - Collector Code: Invalid user.
  - Unim Publish New Thread BH amb Internal Error amb un KO Link.
    - Collector Code: Internal error.

7.5.1.15 Register user BH

- Creem una *Module Unit* (Register User BH) del mòdul Register User BH.
  - Unim Register User BH amb Index BH amb un OK Link.
  - Unim Register User BH amb Error message de Create User amb un Transport Link.
    - Coupling:

        - errorDescription amb errorMsg.
    - Unim Register User BH amb Create User BH amb un KO Link.
        - Collector Code: Register KO.
    - Unim Register User BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.16 Remove Favorite BH

- Creem una *Module Unit* (Remove Favorite BH) del mòdul Remove Favorite BH.
    - Unim Remove Favorite BH amb Thread Info BH amb un OK Link.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Remove Favorite BH amb Index BH amb un KO Link.
        - Collector Code: Invalid thread.
    - Unim Remove Favorite BH amb Login amb un KO Link.
        - Collector Code: Invalid user.
    - Unim Remove Favorite BH amb Internal Error amb un KO Link.
        - Collector Code: Internal error.

7.5.1.17 Thread Info BH

- Creem una *Module Unit* (Thread Info BH) del mòdul Thread Info BH.
    - Unim Thread Info BH amb Thread Info amb un OK Link.
    - Unim Thread Info BH amb Thread de Thread Info amb un Transport Link.
        - Coupling:
            - oidThread amb KeyCondition[oidThread].
    - Unim Thread Info BH amb Responses de Thread Info amb un Transport Link.
        - Collector Code: Thread Info OK.
        - Coupling:
            - oidThread amb l'*Attribute Condition*.
    - Unim Thread Info BH amb Index BH amb un KO Link.

7.5.1.18 User Info BH

- Creem una *Module Unit* (User Info BH) del mòdul User Profile BH.
    - Unim User Profile BH amb User Profile amb un OK Link.
    - Unim User Profile BH amb User Info de User Profile amb un Transport Link.
        - Collector Code: User Profile OK.
        - Coupling:
            - oidUser amb la KeyCondition[oidUser].
    - Unim User Profile BH amb Threads de User Profile amb un Transport Link.
        - Collector Code: User Profile OK.
        - Coupling:
            - oidUser amb l'*Attribute Condition*.
    - Unim User Profile BH amb Responses de User Profile amb un Transport Link.
        - Collector Code: User Profile OK.
        - Coupling:
            - oidUser amb l'*Attribute Condition*.

- Unim User Profile BH amb Favorites de User Profile amb un Transport Link.
  - Collector Code: User Profile OK.
  - Coupling:
    - oidUser amb l'*Attribute Condition*.
- Unim User Profile BH amb Index BH amb un KO Link.

7.5.2 Pàgines

Enllaçarem cada pàgina amb els *Business Helper* amb els links que tenim definits al UX Model i haurem acabat la lògica de l'aplicació.

7.5.2.1 Create Response

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Response Form amb Publish New Response BH amb un Normal Link.
        - Name: Create response.
        - Coupling:
            - Content amb content.
            - oidThread amb oidThread.
    - Unim Create Response amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Create Response amb Logout amb un Normal Link.
        - Name: Logout.

7.5.2.2 Create Thread

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Thread Form amb Publish New Thread BH amb un Normal Link.
        - Name: Create thread.
        - Coupling:
            - Content amb content.
            - Content Type amb content_type.
            - Specialization amb specialization.
            - Title amb title.
    - Unim Create Thread amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Create Thread amb Logout amb un Normal Link.
        - Name: Logout.

7.5.2.3 Create User

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Thread Form amb Publish New Thread BH amb un Normal Link.
        - Name: Create thread.
        - Coupling:
            - Email amb email.
            - Password amb password.
            - Specialization amb specialization.
            - Username amb username.
    - Unim Create User amb Login amb un Normal Link.
        - Name: Login.

7.5.2.4 Index

- Fem les unions necessàries per als links bàsics.
    - Unim Index amb Create Thread BH amb un Normal Link.
        - Name: Create thread.
    - Unim Index amb Modify User BH amb un Normal Link.
        - Name: Modify my profile.
    - Unim Index amb Logout amb un Normal Link.
        - Name: Logout.
- Fem les unions necessàries per als links de cada una de les instàncies mostrades a les llistes.
    - Unim Matching amb Thread Info BH amb un Normal Link.
        - Name: Read thread.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Matching amb User Profile BH amb un Normal Link.
        - Name: Author's profile.
        - Coupling:
            - PostedBy.oidUser amb oidUser.
    - Unim Favorites amb Thread Info BH amb un Normal Link.
        - Name: Read thread.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Favorites amb User Profile BH amb un Normal Link.
        - Name: Author's profile.
        - Coupling:
            - PostedBy.oidUser amb oidUser.
    - Unim Favorites amb Remove Favorite BH amb un Normal Link.
        - Name: Unfavorite.
        - Coupling:
            - oidThread amb oidThread.

7.5.2.5 Internal Error

No té cap unió a altres pàgines, és només un lloc d'entrada.

7.5.2.6 Login

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Login Form amb Do Login BH amb un Normal Link.
        - Name: Login.
        - Coupling:
            - Password amb password.
            - Username amb username.
    - Unim Login amb Create User BH amb un Normal Link.
        - Name: Create new user.

7.5.2.7 Modify Response

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Modify Response Form amb Do Response Modification BH amb un Normal Link.
        - Name: Modify response.
        - Coupling:
            - Content amb content.
            - oidResponse amb oidResponse
    - Unim Modify Response amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Modify Response amb Logout amb un Normal Link.
        - Name: Logout.

7.5.2.8 Modify Thread

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Modify Thread Form amb Do Thread Modification BH amb un Normal Link.
        - Name: Modify thread.
        - Coupling:
            - Content amb content.
            - Content Type amb content_type.
            - oidThread amb oidThread.
            - Specialization amb specialization.
            - Title amb title.
    - Unim Modify Thread amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Modify Thread amb Logout amb un Normal Link.
        - Name: Logout.

7.5.2.9 Modify User

- Fem les unions necessàries per al *Submit* del formulari i els links bàsics.
    - Unim Modify User Form amb Do User Modification BH amb un Normal Link.
        - Name: Modify user.
        - Coupling:
            - Email amb email.
            - New password amb newPassword.
            - oidUser amb oidUser.
            - Current password amb oldPassword.
    - Unim Modify User amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Modify User amb Logout amb un Normal Link.
        - Name: Logout.

7.5.2.10 Thread Info

- Fem les unions per als links bàsics.

    - Unim Thread Info amb Index BH amb un Normal Link.
        - Name: Index.
    - Unim Thread Info amb Logout amb un Normal Link.
        - Name: Logout.
- Fem les unions necessàries per a la *Data Unit* del tema de discussió.
    - Unim Thread amb User Profile BH amb un Normal Link.
        - Name: Author's profile.
        - Coupling:
            - PostedBy.oidUser amb oidUser.
    - Unim Thread amb Response Form de Create Response amb un Normal Link.
        - Name: Create a response.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Thread amb Add Favorite BH amb un Normal Link.
        - Name: Favorite.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Thread amb Remove Favorite BH amb un Normal Link.
        - Name: Unfavorite.
        - Coupling:
            - oidThread amb oidThread.
    - Unim Thread amb Modify Thread BH amb un Normal Link.
        - Name: Modify thread.
        - Coupling:
            - oidThread amb oidThread.
- Fem les unions necessàries per a l'*Index Unit* de les respostes.
    - Unim Responses amb User Profile BH amb un Normal Link.
        - Name: Author's profile.
        - Coupling:
            - FromUser.oidUser amb oidUser.
    - Unim Responses amb Modify Response BH amb un Normal Link.
        - Name: Modify response.
        - Coupling:
            - oidResponse amb oidResponse.
    - Modifiquem el KO Link entre Modify Response BH i Thread Info BH.
        - Coupling:
            - FromThread.oidThread_PASSING amb oidThread.

7.5.2.11 User Profile

- Fem les unions necessàries per als links bàsics.
    - Unim User Profile amb Modify User BH amb un Normal Link.
        - Name: Modify my profile.
    - Unim User Profile amb Index BH amb un Normal Link.
        - Name: Index.

- Unim User Profile amb Logout amb un Normal Link.
  - Name: Logout.

- Fem les unions necessàries per als links de cada una de les instàncies mostrades a les llistes.
  - Unim Threads amb Thread Info BH amb un Normal Link.
    - Name: Read thread.
    - Coupling:
      - oidThread amb oidThread.
  - Unim Responses amb Thread Info BH amb un Normal Link.
    - Name: Read thread.
    - Coupling:
      - FromThread.oidThread amb oidThread.
  - Unim Responses amb User Profile BH amb un Normal Link.
    - Name: Author's profile.
    - Coupling:
      - FromThread.PostedBy.oidThread amb oidThread.
  - Unim Favorites amb Thread Info BH amb un Normal Link.
    - Name: Read thread.
    - Coupling:
      - oidThread amb oidThread.
  - Unim Favorites amb User Profile BH amb un Normal Link.
    - Name: Author's profile.
    - Coupling:
      - oidThread amb oidThread.

7.6 Errors comuns

Durant el transcurs de tot el projecte he tingut molts problemes amb WebRatio que s'han repetit més d'una vegada i que m'han costat de trobar.

Algunes vegades han estat *couplings* de links mal fets o condicions mal fetes però d'altres són més complicats de veure i per això he cregut important explicar-los en aquest apartat i algunes possibles solucions o prevencions.

7.6.1 Valors *null* a les *Math Unit*

Les unitats matemàtiques no accepten valors *null*.

Durant l'execució de l'aplicació es pot observar que els links no funcionen si utilitzen una unitat amb aquest problema. L'execució d'una unitat matemàtica amb *null* provoca un error als *logs* de l'aplicació.

Per a veure que realment aquest és el problema només cal anar als *logs* i veure que s'ha afegit un amb el problema de l'ús d'un valor *null* a l'unitat matemàtica.

Els logs de l'aplicació estan al Tomcat dins la carpeta de l'aplicació a /WEB-INF/log i cal obrir el fitxer RTXError.log. L'error seria com el següent:

```
03 ene 2013 00:01:33,859  ERROR [http-8080-1]
(com.webratio.units.utility.rtx.MathUnitService_$$_javassist_15:201) -
[2B8DCAF4A6C42B59F8B974136DC8DB97][miu21][opm25][mathu4]Could not
execute service

 java.lang.ArithmeticException: = : the second operand cannot be null
```

Hauriem de comprovar que l'error s'ha donat en el moment actual i després observar en quina unitat es dona. Els noms són els identificadors de la unitat i no els seus noms. Això ens obliga a buscar quina és però pels noms podem saber una mica més de la seva localització.

En aquest cas, *miu* fa referència a una *Module Unit*, *opm* a un *Operation Module* i *mathu* és la *Math Unit* que ens està donant el problema.

Per tal de tenir això en compte, sempre és convenient usar una *Is Not Null Unit* abans d'executar-la per a cada un dels operands.

7.6.2 KO links als *OK Collector*

Un altre problema del que em va costar trobar la font va ser aquest.

Resulta que als *OK Collector* no accepten tenir com a entrada un KO Link o com a mínim l'execució quan es donava el cas no era la correcte. Aquest cas es veu agreujat perquè en cap moment l'*OK Collector* denega que pugui ser destí d'un KO Link a WebRatio.

Per tal de solucionar-ho, es pot incorporar una *No Op Operation Unit* on fer arribar el KO Link i després unir aquesta unitat amb el *OK Collector* amb un OK Link.

En el cas de necessitar passar paràmetres amb el Link, podríem fer-ho si els passem per PASSING fins la unitat i després els enllacem amb l'altre linik fins l'*OK Collector*.

7.6.3 Custom descriptors

L'ús dels C*ustom descriptor* poden alterar el comportament d'algunes unitats. Això acostuma a passar quan s'ha usat un *Custom descriptor* en algun moment o s'ha habilitat l'opció en una unitat tot i no utilitzar-lo després.

A continuació explico alguns dels casos que més maldecaps han provocat i una forma de veure si s'està donant el problema amb una possible solució.

- **Pàgines**
  Quan s'ha usat un *Custom descriptor* en una pàgina pot ser que els canvis que fem no es vegin quan executem l'aplicació. Acostumen a ser noves unitats que no teníem abans i que no es mostren.
- **Unitats**
  Quan el problema es dona en una unitat, el que acostuma a passar és que l'execució no funciona com hauria. Algunes vegades acaba amb uns valors que no són els esperats o torna enrere sense motiu aparent.

Destacar que moltes vegades, si es veu algun d'aquests problemes comentats, es pot veure si hi ha algun *Custom descriptor* que no s'està usant si es fa servir *Find > Find model problems* dins la vista on trobem el problema. A l'apartat de *warnings* del resultat indica que hi ha un *Custom descriptor* sense utilitzar a la unitat en qüestió.

La solució pot ser canviar el nom de la unitat a un nom que no tingués el *Custom descriptor* o esborrar el *custom descriptor* de la carpeta /WEB-INF/descr del projecte en qüestió. Això és donat a que els *custom descriptors* estan associats al nom de la unitat i al repetir el mateix identificador dona aquests problemes.

Un cop utilitzat el *Custom decsriptor*, WebRatio congela l'arxiu de forma que des d'aquell moment tu ets el responsable d'escriure manualment el comportament que tindrà la unitat a la que descriu. És una funcionalitat que s'ha d'usar rara vegada però que al estar tocant la configuració per a aprendre el seu funcionament pot donar aquest tipus de maldecaps.

8 Patrons de traducció

Un cop traduïda l'aplicació d'UML a WebML ens centrarem en aquest apartat a extreure possibles patrons de traducció d'un model a un altre.

8.1 Com són els patrons?

Més que patrons hauríem d'anomenar-los *best practices*.

Els patrons són una sèrie de recomanacions on donat un fragment d'UML o una estructura, explica una possible solució a WebML de manera que l'usuari pugi traduir d'un a l'altre partint d'una base sòlida.

Així doncs, per exemple, podem ajudar al traductor a modelar el web des del diagrama intern cap a una estructura de pàgines i mòduls a WebML o a traduir d'un fragment condicional d'un diagrama de seqüència a una sèrie d'operacions de WebML.

8.2 Metodologia d'extracció

Per tal de trobar aquests possibles patrons haurem de trobar coses en comú entre diferents parts del models, tant a l'UML com al WebML.

El primer de tot és observar l'estructura més general donada per l'UX Model i el diagrama intern amb l'extensió web ja que sembla el més propens a poder tenir un patró al ser específic de les aplicacions web.

Un cop puguem tenir definida una estructura caldrà omplir cada una de les parts amb la traducció directe de les operacions i els diagrames UML. Revisant totes les traduccions que hem documentat a WebML, podem veure comportaments idèntics en algunes operacions. Aquests comportaments són els que mirem de veure quin era el seu model d'origen per a veure quin comportament a l'UML té traduccions semblants.

A més de tot això, s'ha afegit alguna traducció experimental d'aspectes que el projecte no ha abastat com els *loops* o tipus concret de camps de formulari.

8.3 Patrons

Durant l'extracció de patrons s'han trobat de dos diferents tipus: els estructurals i els operacionals.

Anomenem **estructurals** a aquells patrons que indiquen una forma d'organitzar el web o les operacions però que s'abstrauen del contingut d'aquests. Usualment, dins d'un patró estructural hi ha espais o seccions que s'hauran d'omplir amb patrons operacionals per al seu correcte funcionament.

Els patrons **operacionals**, per altra banda, són patrons molt més concrets a nivell de diagrames de seqüència i permeten fer una traducció gairebé directe de l'operació d'UML a WebML.

A continuació s'explicaran els diferents patrons que s'han extret amb les corresponents explicacions per al seu ús.

8.3.1 Conceptes previs

Per tal d'arribar a entendre els patrons s'ha de conèixer una mica el funcionament de WebML.

Com gairebé tots els patrons s'han extret de la traducció del fòrum, estan basats en la forma en la que he entès jo el disseny en WebML.

Per posar un exemple, he intentat usar Transport Links sempre que hagués de passar informació entre unitats que no estan unides amb un OK o KO Link en comptes d'anar passant la informació d'unitat en unitat perquè em va semblar més canviable.

En alguns casos, hi ha moments que és necessari utilitzar les *Script Unit*. Tal com ja s'ha vist, aquestes estan definides a partir del llenguatge Groovy i en cas de voler fer operacions una mica complexes s'ha de consultar el seu manual per a fer-ho correctament ja que no s'explica aquí tota la seva potència.

8.3.2 Operacionals

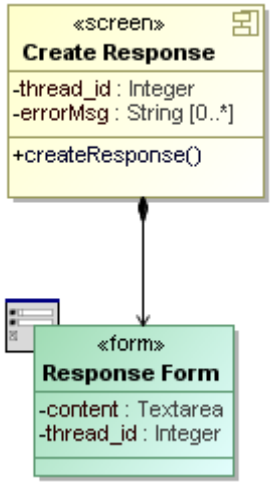
8.3.2.1 D'screens d'UX Model a pàgines de WebML

A l'UX Model representem la informació que volem mostrar a una pàgina concreta.

Aquest patró està a l'apartat de patrons operacionals tot i que parli d'una estructura com és la pàgina ja que explicarà com traduir bloc a bloc les possibles estructures de les que es composa una pàgina.

Formularis

Un dels elements més usats en aquest projecte és el formulari. A partir del formulari permetem que l'usuari pugui introduir dades per a guardar al sistema en forma de temes de discussió o respostes.



Il·lustració 75: Formulari a una pàgina (UML)

En tots els casos i sense excepcions, la traducció s'ha fet amb una Entry Unit. A aquesta unitat se li han afegit els camps necessaris segons el tipus d'entrada que desitjàvem.

1. Creem una *Entry Unit* amb el nom del formulari dins la pàgina que el conté.
2. Afegim els camps del formulari del tipus que correspongui amb els paràmetres necessaris per al seu funcionament.
3. Afegim els filtres en els camps segons sigui necessari si volem restringir els valors dels camps.
4. Reordenem els camps amb l'ordre que volem que es mostrin a l'usuari.
5. Unim el formulari amb un Normal Link amb l'operació que gestioni la informació introduïda per a formar el botó de *submit*. El nom del link serà el nom del botó.

Hi ha tres tipus de camps possibles a WebML. El primer són els *Fields* que corresponen a camps de text, *radio button* o pujada d'arxius, el segon són els *Selection Fields* que són els *combo box* i l'últim els *Multi Selection Fields* que permeten crear una o més *checkbox*.

A continuació s'explica cada tipus de camp com s'ha de configurar per tal d'afegir-lo al formulari.

Field

1. El *Field* crearà un camp per a omplir per l'usuari.
    a. Name: és el nom que tindrà el camp un cop representat.
    b. Type: el tipus de la variable que esperem d'entrada al camp.
    Les opcions *date*, *time* i *timestamp* afegeixen un calendari per a la selecció del valor.
    Boolean genera dos *radio button* amb les opcions *true* i *false*.
    Blob genera la opció de pujar arxius amb la possibilitat d'afegir-hi un filtre pel tipus d'arxiu.
    c. Preloaded: en el cas de estar marcat és possible donar-li valor a través d'un link (un Transport Link és recomanable).
    d. Modifiable: indica si l'usuari podrà o no modificar el camp. Semblant al *enabled* o *disabled* a l'HTML.
    e. Hidden: indica si és un camp ocult del formulari de tal manera que l'usuari no el podrà veure perquè no es mostrarà. Si està marcat no té sentit que el camp sigui *Modifiable* i acostuma a ser *Preloaded*.

Selection Field

1. El *Selection Field* permet a l'usuari escollir un element d'una llista.
    a. Name: és el nom que tindrà el camp un cop representat.
    b. Type: el tipus de la variable que esperem mostrar.
    c. Preselected Value: valor marcat per defecte en el cas de saber ja quins són els possibles i vulguem sempre tenir el mateix. Es pot omplir a través de links.
2. Per tal de definir la llista de possibles valors a escollir, generem un *Slot* relacionat amb el *Selection Field*.
    a. Name: és el nom que es veurà al intentar fer enllaços amb els camps del formulari.
    b. Value: introduïm els valors possibles separats per | si els valors són determinats. És possible reescriure aquests valors o assignar-los més tard a partir d'enllaços.
    c. Label: indica si els valors de l'*Slot* són els que es mostraran. Cal que estigui marcat.
    d. Output: indica si els valors de l'*Slot* són els que s'enviaran al acceptar el formulari. Hauria d'estar desmarcat.
3. Per tal de definir la llista de valors que s'enviaran al gestor de la informació, generem un *Slot* relacionat amb el *Selection Field*.
    a. Name: és el nom que es veurà al intentar fer enllaços amb els camps del formulari.
    b. Value: introduïm els valors possibles separats per | si els valors són determinats. És possible reescriure aquests valors o assignar-los més tard a partir d'enllaços. Han d'estar en el mateix ordre que els seus corresponents del punt anterior.

c. Label: indica si els valors de l'*Slot* són els que es mostraran. Hauria d'estar desmarcat.
d. Output: indica si els valors de l'*Slot* són els que s'enviaran al acceptar el formulari. Cal que estigui marcat.

En el cas de voler rebre al gestor de la informació el mateix que mostrem a l'usuari, podem marcar *Label* i *Output* en un mateix *Slot* i eliminar l'altre.

En el cas de generar la llista a partir de més d'una llista original, podem fer la concatenació utilitzant diferents *Slot* i omplint-los per separat. Usant l'atribut *Slot Order* del *Selection Field* podem escollir l'ordre en que es concatenarien.

Multi Selection Field

1. El *Selection Field* permet a l'usuari escollir varis elements d'una llista a partir de *checkboxes*.
    a. Name: és el nom que tindrà el camp un cop representat.
    b. Type: el tipus de la variable que esperem mostrar.
    c. Preselected Value: valor marcat per defecte en el cas de saber ja quins són els possibles i vulguem sempre tenir el mateix. Es pot omplir a través de links.
2. Per tal de definir la llista de possibles valors a escollir, generem un *Slot* relacionat amb el *Multi Selection Field*.
    a. Name: és el nom que es veurà al intentar fer enllaços amb els camps del formulari.
    b. Value: introduïm els valors possibles separats per | si els valors són determinats. És possible reescriure aquests valors o assignar-los més tard a partir d'enllaços.
    c. Label: indica si els valors de l'*Slot* són els que es mostraran. Cal que estigui marcat.
    d. Output: indica si els valors de l'*Slot* són els que s'enviaran al acceptar el formulari. Hauria d'estar desmarcat.
3. Per tal de definir la llista de valors que s'enviaran al gestor de la informació, generem un *Slot* relacionat amb el *Multi Selection Field*.
    a. Name: és el nom que es veurà al intentar fer enllaços amb els camps del formulari.
    b. Value: introduïm els valors possibles separats per | si els valors són determinats. És possible reescriure aquests valors o assignar-los més tard a partir d'enllaços. Han d'estar en el mateix ordre que els seus corresponents del punt anterior.
    c. Label: indica si els valors de l'*Slot* són els que es mostraran. Hauria d'estar desmarcat.
    d. Output: indica si els valors de l'*Slot* són els que s'enviaran al acceptar el formulari. Cal que estigui marcat.

En el cas de voler rebre al gestor de la informació el mateix que mostrem a l'usuari, podem marcar *Label* i *Output* en un mateix *Slot* i eliminar l'altre.

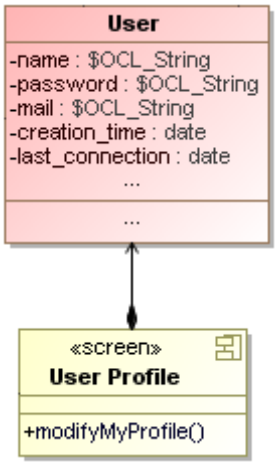
En el cas de generar la llista a partir de més d'una llista original, podem fer la concatenació utilitzant diferents *Slot* i omplint-los per separat. Usant l'atribut *Slot Order* del *Selection Field* podem escollir l'ordre en que es concatenarien.

En el cas de les validacions, hem vist que és possible fer que un camp sigui obligat o que tingui la forma d'un e-mail però hi ha moltes altres opcions. Tot i així, s'ha de tenir en compte que aquestes comprovacions no són un substitut de les comprovacions a domini ja que podrien ser modificades en algun moment i eliminar la restricció a presentació i no ser consistent el sistema.

No s'explicaran tots els tipus de restriccions perquè està cada un documentat al seu apartat a la guia de WebML però cal recordar que per a cada un és recomanat escriure el missatge d'error que es mostrarà a l'usuari. En cas de no estar definit l'usuari només veurà el nom de la restricció.

Informació d'una instància

Quan definim els UX Model, a vegades es creen associacions amb la pàgina quan volem mostrar certa informació d'una classe. En el cas de que la relació sigui de cardinalitat 1, hem utilitzat una *Data Unit* per a mostrar aquesta informació.



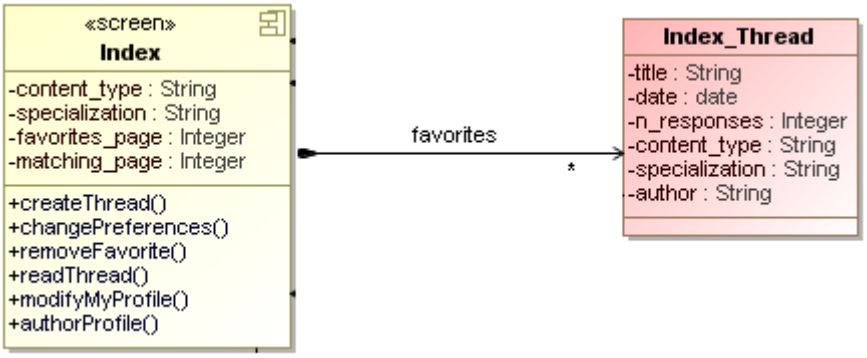
Il·lustració 76: Informació d'una instància a una pàgina (UML)

La *Data Unit* permet escollir els camps entre els atributs d'una classe i les importacions de relacions. Per a utilitzar-la només hem de definir quin tipus d'entitat representa i assignar l'identificador corresponent a la instància que volem mostrar.

1. Creem una *Data Unit* dins la pàgina de l'entitat de la qual volem mostrar la informació.
2. Escollim la informació que volem mostrar d'aquesta entitat.
3. Escollim l'ordre en que es mostrarà la informació que hem escollit.
4. Enllacem l'origen de l'identificador amb la unitat amb un Transport Link per tal de que mostri la informació correcte.

Informació de múltiples instàncies

Quan definim els UX Model, a vegades es creen associacions amb la pàgina quan volem mostrar certa informació d'una classe. En el cas de que la relació sigui de cardinalitat N, hem utilitzat una *Index Unit* per a mostrar aquesta informació.



Il·lustració 77: Informació de múltiples instàncies a una pàgina (UML)

La *Index Unit* permet mostrar informació de múltiples instàncies donada l'entitat a la que representen. A partir de *Key Condition* podem restringir la informació a un conjunt d'identificadors i a partir de *Attribute Condition* podem generar *queries* per a escollir les instàncies que ens interessin. En el cas de que ens interessi, també podem restringir el màxim d'instàncies a mostrar.

1. Creem una *Index Unit* dins la pàgina de l'entitat de la qual volem mostrar la informació.
2. Escollim la informació que volem mostrar d'aquesta entitat per a totes les instàncies.
3. Afegim les *Key Condition* o *Attribute Condition* en el cas de que sigui necessari per a fer la selecció d'instàncies a mostrar.
4. Enllacem els orígens dels valors per a les condicions amb la unitat amb Transport Links per tal de que es mostri la informació correcte.

En el cas concret de voler tenir paginació dins la informació que mostrem, és possible usar la *Power Index Unit*. El seu funcionament és molt semblant a l'*Index Unit* convencional però ofereix el paràmetre *Block Factor* que permet dir el nombre màxim d'instàncies per pàgina i ja ofereix a l'usuari els links per a canviar de pàgina típics de la paginació.

Informació general

En el cas de voler mostrar certa informació que només aporta contingut (missatges d'error, missatges predeterminats o informació concreta) podem usar la *Multi Message Unit*.

Il·lustració 78: Informació general (errorMsg) (UML)

La *Multi Message Unit* permet escriure un text i deixar *placeholders* per a poder omplir-los més tard mitjançant links i d'aquesta forma poder mostrar text amb informació de l'aplicació.

1. Creem una *Multi Message unit* dins la pàgina.
2. Modifiquem el *Message Text* de la unitat introduint el text que volem mostrar i els *placeholders* entre dos símbols $ amb el nom que voldrem veure al fer la unió.
3. Unim els orígens dels valors a mostrar amb la unitat amb Transport Links.

8.3.2.2 Excepcions

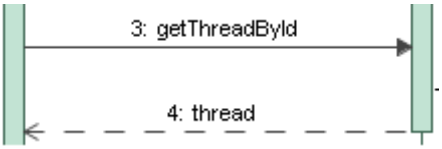
Quan parlem d'excepcions parlem acostumem a referir-nos a errors en el comportament.

En aquest cas, en WebML tenim els KO links que acostumen a ser sortides d'unitats en el cas de trobar algun error o un comportament no desitjat.

Base de dades

Com a exemple podem veure el cas de les transaccions que han d'acabar en el cas de que un dels paràmetres no sigui correcte.

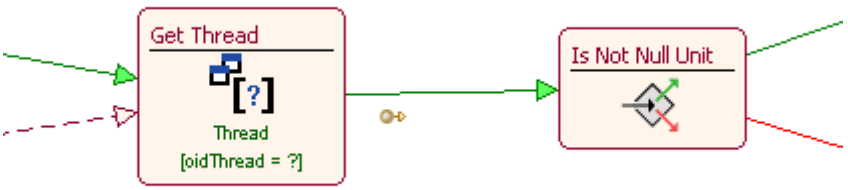
En el cas concret de la transacció de crear un nou tema (TxGetThreadInfo) intentem accedir a la base de dades amb l'identificador del tema de discussió per a extreure la informació relacionada.



Il·lustració 79: Extracció de base de dades amb excepció (UML)

En el nostre disseny, en UML esperaríem que el controlador de la base de dades ens contestés amb una excepció en el cas de no existir cap tupla amb l'identificador que hem passat, però a WebRatio rebem un resultat *null* al OK Link de la unitat de selecció si això passa però cap error.

Per tant, el que faríem es passar l'identificador del resultat per una *Is Not Null Unit* i utilitzar el KO Link de sortida de la unitat per a indicar que hi ha hagut un comportament no desitjat i aturar la transacció.

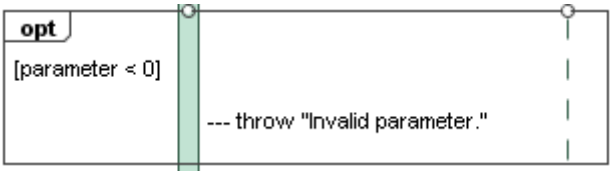


Il·lustració 80: Extracció de base de dades amb excepció (WebML)

En el moment de crear la nova instància de Thread pot ser que es produeixi algun problema. En aquest cas ja no és necessari fer servir la *Is Not Null Unit* perquè la unitat de creació ja té una sortida de KO Link per a indicar els possibles errors d'inserció.

Condicions

Altres vegades volem llançar excepcions en el cas de que certa condició no es compleixi (paràmetres incorrectes) però sense cap accés a la base de dades. Normalment ho solucionem amb una sentència condicional *(IF)*.

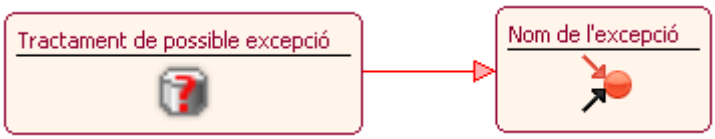


Il·lustració 81: Llançament d'excepció per a comprovació de paràmetres (UML)

Per a aquests tipus de casos no és el KO Link la solució ja que estem utilitzant una unitat de *switch* per a diferenciar els diferents resultats i aquesta només té OK Links, un per a cada possible resultat identificat i un altre per a quan no es compleix cap de les expectatives. S'entra en més detall sobre les operacions de control de flux en el patró operacional "Control de flux (condicionals i bucles)".

Normalment aquesta operació s'executa dins d'un mòdul ja que hem decidit representar els *business helper* i les transaccions d'aquesta manera. Per tal de poder tractar l'excepció des de fora en aquests casos, enllacem la *Switch Unit* amb un OK Link amb el *code* corresponent cap a un *KO Collector* fent que des de fora ja puguin tractar la excepció amb un KO Link amb el codi del *KO Collector*.

Per a resumir, en el cas de voler tractar una excepció d'aquest tipus hem d'avaluar la condició segons el patró corresponent i usar un *KO Collector* per tal de poder-la tractar.



Il·lustració 82: Llançament d'excepció per a comprovació de paràmetres (WebML)

La unitat amb l'interrogant la utilitzem en els exemples com a *placeholder* per a un conjunt operacions.

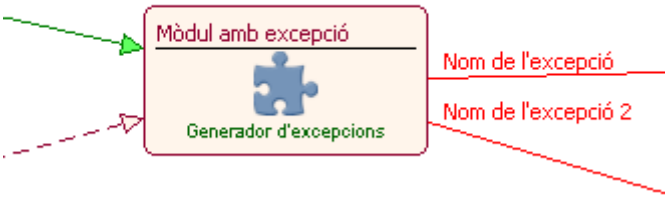
Propagació d'excepcions dins d'un mòdul

Quan ens trobem dins d'un mòdul, és possible utilitzar les unitats *OK Collector* i *KO Collector* que identificaran diverses sortides per a la instància del mòdul.

En el cas de que vulguem propagar alguna de les excepcions cap a fora, hauríem d'utilitzar una unitat *KO Collector* i ajuntar l'origen de l'excepció amb aquesta unitat.

L'ús des de fora és molt simple. Un cop tinguem instanciat el mòdul a utilitzar, podrem utilitzar un KO Link com a sortida i escollir a quin *KO Collector* fem referència (*Collector Code*) i fins i tot

passar paràmetres si els hem assignat. Podem usar tants *KO Collectors* com siguin necessaris i identificar-los amb un nom per a saber en tot moment a quina excepció fan referència.



Il·lustració 83: Captura d'excepcions d'una operació (WebML)

Si és necessari passar cap paràmetre a l'excepció es poden afegir els *Output Collector Parameter* als *KO Collector* per a accedir a aquesta informació des de fora en el cas de que es llanci l'excepció.

Tractament de les excepcions

Pàgina d'error

Un dels casos més comuns és utilitzar la informació de l'excepció per a mostrar un missatge d'error o simplement mostrar una pàgina diferent a la que mostraries en cas d'èxit.

Per a fer això, només cal dirigir el link cap a la pàgina o a la unitat que vulguem mostrar i passar els paràmetres si els tenim com si fos un link qualsevol.

Seguir amb la cadena d'operacions per una altra banda tractant l'error

En el cas de que amb l'excepció vulguem executar una sèrie diferents d'operacions, per a tractar l'error, per exemple, l'utilitzaríem com el OK Link al enllaçar operacions però sent un KO Link en aquest cas. Això vol dir que les operacions seguiran la cadena segons el resultat d'aquella unitat.

8.3.2.3 Operacions aritmeticològiques

Les operacions aritmeticològiques són operacions que acostumem a fer per a obtenir un valor a partir d'altres ja coneguts o dins de sentències d'operacions condicionals.

Dins del projecte s'ha usat sempre les *Math Unit* per a les sentències d'operacions condicionals. Tot i així, s'explicarà a més a més la possibilitat que ofereixen les *Script Unit* per a poder fer operacions més potents sense necessitat de fer diagrames de gran tamany.

Destacar que dins d'aquestes comparacions no inclourem les comparacions amb el valor *null* ja que per a això ja existeix una unitat específica, la *Is Not Null Unit*.

Math Unit

La *Math unit* és capaç de fer tot un seguit d'operacions aritmeticològiques sobre uns operands prèviament definits i una expressió que pot ser modificada com a paràmetres.

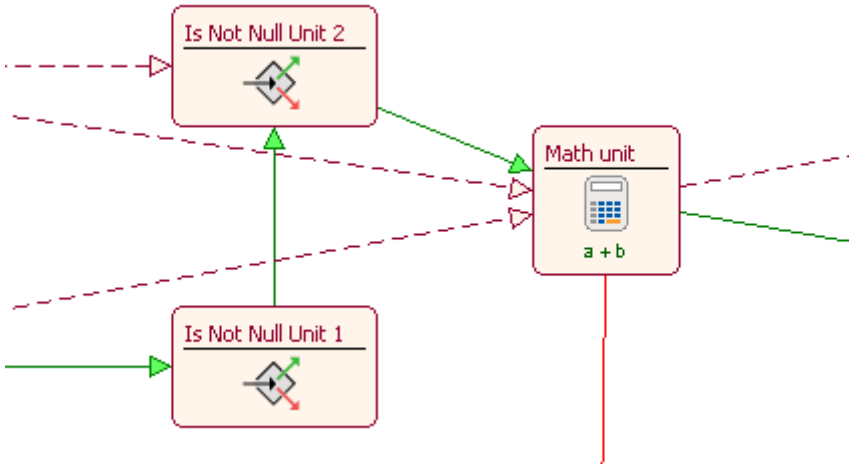
Per defecte, el seu ús podria ser el següent:

1. Creem una *Math Unit*.
2. Afegim un *Operand* per a cada operand que necessitem per a fer l'operació aritmeticològica.
3. Escollim el tipus de resultat que volem que retorni la unitat (boolean, float, integer o decimal).
4. Definim la *Default Expression* utilitzant els operands i els operadors disponibles.
5. Enllacem les variables amb els operands corresponents amb Transport Links.
6. Unim l'operació anterior amb la *Math Unit* amb un OK Link.
7. Unim la *Math Unit* amb la següent operació amb un OK Link.
8. Enllacem el resultat de la *Math Unit* amb un Transport Link. El nom de la variable resultant és Result.
9. Unim la *Script Unit* amb la següent operació que vulguem fer en cas d'error (normalment un *KO Collector* d'error intern dins d'un mòdul) amb un KO Link.

S'ha de vigilar sempre de no utilitzar la unitat amb valors *null* com a operands d'entrada perquè provoca errors d'execució. És recomanable sempre utilitzar abans una *Is Not Null Unit* per no executar la unitat si algun valor no era l'esperat. Una altra opció és utilitzar el KO Link des de la unitat per tractar aquest error.

Les operacions capaç de fer la unitat estan a l'especificació de la mateixa unitat. Entre elles trobem les operacions aritmeticològiques bàsiques i algunes funcions matemàtiques com arrels, exponencial, logaritmes, negatiu, funcions trigonomètriques i una funció aleatòria.

A la imatge mostrada a continuació podem veure el diagrama WebML d'un exemple que fa la suma de dos operands.



Il·lustració 84: Operacions aritmeticològiques amb Math Unit (WebML)

Script Unit

La *Script Unit* és una de les unitats més potents de les que suporta WebML. Dins d'aquesta unitat podem especificar un tros de codi en llenguatge Groovy [1].

Aquesta potència ens pot permetre no només fer operacions aritmètiques fent servir funcions pròpies sinó eliminar el tractament dels valors *null* amb les *Is Not Null Unit* i tractar-lo dins de la mateixa unitat. Tot i així, s'han de seguir tractant aquests valors perquè donen problemes com la majoria de llenguatges de programació si intentem fer operacions aritmètiques amb ells. No passa el mateix amb les comparacions que l'accepten com un valor més.

A continuació es mostrarà com podem recrear un petit exemple fent servir Groovy per a donar unes nocions bàsiques del llenguatge.

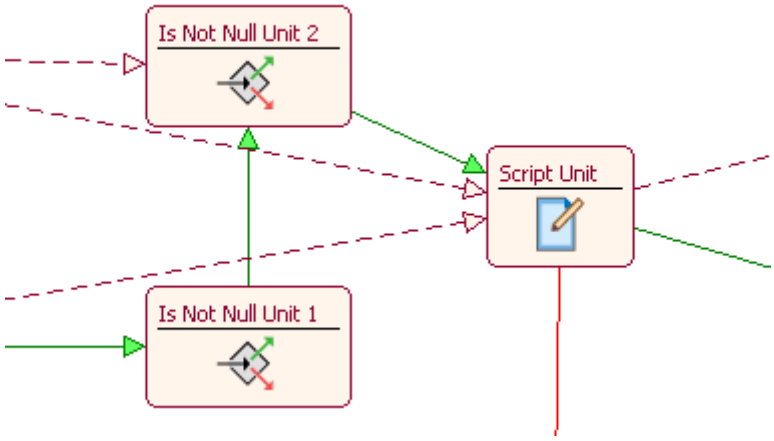
1. Creem una *Script Unit*.
2. Editem l'*Script Text* i afegim el següent codi com a cos de l'*script*.

```
#input Integer a
#input Integer b
#input Integer c
#output Boolean comparison
return [ "comparison" : a + b == c ]
```

3. Enllacem els valors amb la *Script Unit* amb un Transport Link.
4. Unim l'operació anterior amb la *Script Unit* amb un OK Link.
5. Unim la *Script Unit* amb la següent operació amb un OK Link.
6. Enllacem el valor de sortida de la *Script Unit* amb les operacions que ho necessitin. El nom de la variable o variables serà el que haguem definit a l'output.
7. Unim la *Script Unit* amb la següent operació que vulguem fer en cas d'error (normalment un *KO Collector* d'error intern dins d'un mòdul) amb un KO Link.

[1] http://groovy.codehaus.org

Tal com es veu, en cas de no comprovar els paràmetres és possible. Igual que a la *Math Unit* cal utilitzar el KO Link des de la unitat per a controlar qualsevol error intern.



Il·lustració 85: Operacions aritmeticològiques amb Script Unit (WebML)

Una de les avantatges respecte la *Math Unit* és que podem definir més d'un valor de retorn i tornar-los separats per comes com si fos un *hash*.

8.3.2.4 Control de flux (condicionals i bucles)

Condicionals

Hi ha vegades durant la definició de l'UML que es necessita utilitzar els fragments d'interacció amb les etiquetes ALT o OPT per a tal d'executar certes accions segons un condicional. Les traduccions a WebML es fan de forma diferent depenent del tipus.

OPT (IF)

Quan utilitzem un OPT volem que certes operacions només es facin en un cas concret i si no es compleix es salti el bloc complet i segueixi l'execució.

Per tal de recrear això a WebML es necessita una unitat que faci de divisora i capaç de generar dos links diferents segons la condició. En un dels casos s'executaran les operacions necessàries i en l'altre s'enllaçarà directament al final d'aquestes.

Com la divisió depèn de la condició de l'OPT, el primer que es fa és computar el resultat de la condició i un cop fet tractar-lo. Pot donar-se el cas de que la condició sigui una comparació amb *null* (comprovar si existeix un paràmetre) o que sigui una comparació d'un altre tipus.

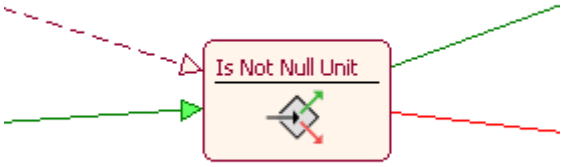
En el primer cas s'utilitza la *Is Not Null Unit*. Aquesta unitat pren per entrada un valor i continua l'execució per l'OK Link en el cas de no ser *null* o pel KO Link en cas de ser-ho.

Podem trobar casos d'aquest tipus en la transacció de la modificació dels temes de discussió (TxModifyThread) on només modifiquem un camp si no és *null*.

Il·lustració 86: OPT. Comprovació de parametre null (UML)

Si intentem resumir d'alguna forma el procés per a la traducció seria el següent.

1. Creem una *Is Not Null Unit*.
2. Enllacem amb un Transport Link la unitat productora del valor amb la *Is Not Null Unit* fent *coupling* del valor amb l'*Input* de la unitat.
3. Unim l'operació anterior amb la *Is Not Null Unit* amb un OK Link.
4. Unim la *Is Not Null Unit* amb la primera operació de l'OPT amb un OK Link.
5. Unim la *Is Not Null Unit* amb la següent operació fora de l'OPT amb un KO Link.



Il·lustració 87: Comprovació de paràmetre null (WebML)

Destacar que això es compleix si la condició és que no sigui *null*. En cas de que la condició sigui *true* en el cas de que sigui *null*, canviaríem l'ordre dels links dels punts 4 i 5 de forma que a la primera operació de l'OPT aniria el KO Link i al contrari.

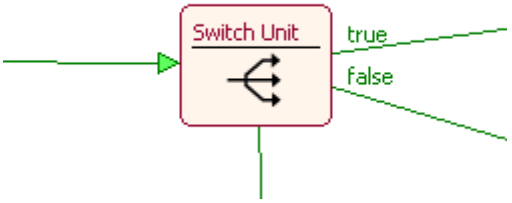
En el segon cas primer hem de computar el resultat de la comparació. Com no és més que el resultat d'una operació lògica, la forma de generar el valor podem escollir-la entre una *Math Unit* o una *Script Unit* segons sigui el cas. El patró d'Operacions aritmeticològiques dona tota la informació sobre aquestes opcions i com s'implementen.

Il·lustració 88: OPT. Condició aritmeticològica (UML)

Un cop ja tenim el valor cal poder dividir l'execució en dues branques segons si el resultat és cert o fals. Per a fer això hem utilitzat la *Switch Unit* per a identificar si el resultat és *true* o *false*. Per a fer-ho afegirem com a possibles *Case Values* els valors *true* i *false*. A partir d'aquest moment, des de la *Switch Unit* podem treure un OK Link per a cada possibilitat.

Amb la divisió ja feta, el comportament és el mateix que en el cas de la comparació amb *null*.

1. Generem el codi necessari per a computar el resultat de la condició segons l'explicat al patró d'Operacions aritmeticològiques.
2. Creem una *Switch Unit* amb *Case Values true* i *false*.
3. Unim el resultat de la computació de la condició amb la *Switch Unit* amb un OK Link enllaçant el paràmetre amb el *Switch* de la unitat.
4. Unim la *Switch Unit* amb la primera operació de l'OPT amb un OK Link amb *Code true*.
5. Unim la *Switch Unit* amb la següent operació fora de l'OPT amb un OK Link amb *Code false*.
6. Utilitzem un OK Link sense *Code* per a tractar una possible excepció si no correspon cap dels dos paràmetres possibles.

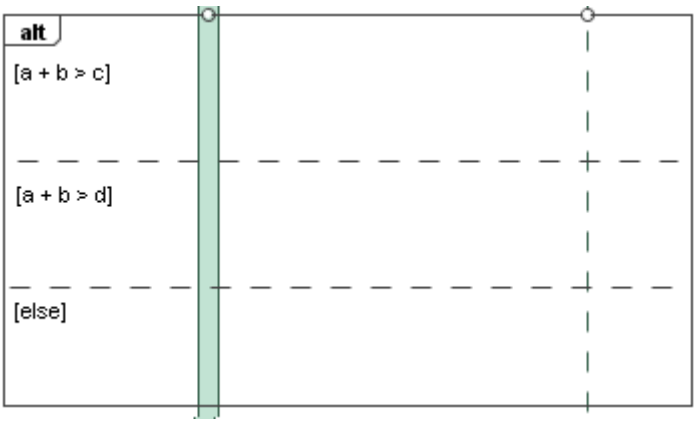


Il·lustració 89: Comprovació d'un resultat booleà (WebML)

En el cas de que la *Switch Unit* no rebi com a entrada cap dels dos valors possibles, l'execució seguirà per a un tercer OK Link sense *Code* (default). En aquests casos és recomanable que aquest link uneixi la unitat amb alguna altra unitat d'error com s'ha vist en el tractament d'excepcions. En el nostre cas, aquest tipus d'operacions formen part dels *business helper* i l'enllacem amb un *KO Collector*.

ALT (IF – ELSE IF – ELSE)

L'ALT s'utilitza quan tenim diferents condicions per a diferents blocs. Podríem dir que són diferents OPT enllaçats que si un es compleix s'executa el seu bloc i ja no cal comprovar la resta i a més a més tindrà una opció final per al cas en que no es compleixi cap d'aquestes condicions anteriors.



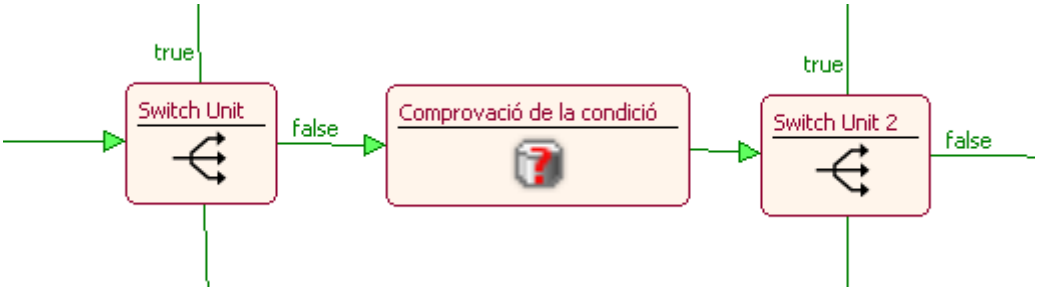
Il·lustració 90: ALT. Multiples comparacions (UML)

Durant el projecte aquest tipus d'especificació no s'ha usat però com està relacionat amb el punt anterior i les unitats necessàries s'han tractat, s'explica una forma teòrica de com es podria traduir.

La primera forma possible és enllaçar diferents OPT tal com hem dit de manera que si no es compleix una condició es comprovi el següent i en cas de complir-se es vagi a la primera operació del bloc. Al final de cada bloc s'ha de tenir en compte que la següent operació serà la primera de fora de tot l'ALT.

A l'exemple següent es tracta amb l'opció de la *Switch Unit*. Si es tracta d'una comparació amb *null* s'hauria de fer amb la *Is Not Null Unit* tal com s'ha vist anteriorment.

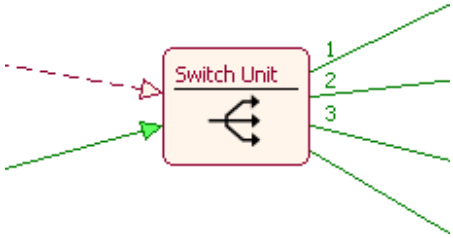
1. Generem el codi necessari per a computar el resultat de la primera condició segons l'explicat al patró d'Operacions aritmeticològiques.
2. Creem una *Switch Unit* amb *Case Values true* i *false*.
3. Unim el resultat de la computació de la condició amb la *Switch Unit* amb un OK Link enllaçant el paràmetre amb el *Switch* de la unitat.
4. Unim la *Switch Unit* amb la primera operació de l'ALT corresponent amb un OK Link amb *Code true*. A partir d'aquí s'escriuran les operacions de dins del bloc.
5. Si hi ha més blocs, repetim el mateix procés tenint en compte com a última operació per a enllaçar la *Switch Unit* i fem l'enllaç amb el *Code false*. En el cas de ser un *else*, utilitzarem el link per a enllaçar amb la primera operació del bloc.
6. Un cop acabats els blocs, unim els finals de cada ALT amb el link que fagi falta a la primera operació de fora dels blocs. Fem el mateix des de l'última comprovació de condició amb un OK Link amb *Code false* si no hi havia *else*.



Il·lustració 91: ALT. Multiples comparacions amb Switch Units (WebML)

Si la comprovació sempre es fa amb el mateix valor i només canvia amb quin valor es compara tenim, una altra forma és tractar-ho és a partir de la *Switch Unit* en el seu estat natural, utilitzant com a entrada l'expressió de l'esquerra i comparant-la amb un valor predeterminat.

1. Generem el codi necessari per a computar el resultat de la part variable de la condició segons l'explicat al patró d'Operacions aritmeticològiques. En cas de fer la comparació amb una variable sense tractar, no caldrà aquest pas.
2. Creem una *Switch Unit* amb un *Case Value* per a cada possible valor de la part variable que vulguem tractar.
3. Unim el resultat de la computació de la part variable de la condició amb la *Switch Unit* amb un OK Link enllaçant el paràmetre amb el *Switch* de la unitat.
4. Unim la *Switch Unit* amb la primera operació de cada ALT amb un OK Link amb *Code* del resultat de l'ALT que enllacem.
5. Unim la *Switch Unit* amb la primera operació del bloc de la condició de *else* si existeix o amb la primera operació de fora el bloc si no amb un OK Link sense *Code*.
6. Unim l'última operació de cada ALT amb la primera operació de fora el bloc amb el link que faci falta.

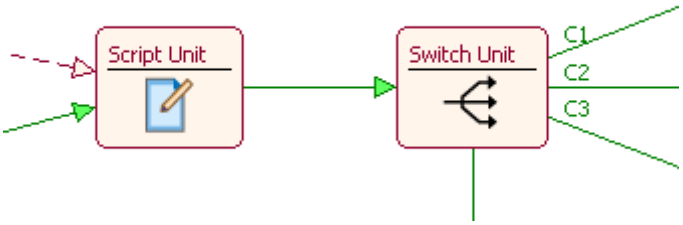


Il·lustració 92: ALT. Multiples possibles resultats amb Switch Unit (WebML)

A la imatge podem veure un cas on distingim els valors 1, 2, 3 i la resta.

Per acabar, podem utilitzar una *Script Unit* per a retornar un codi de quin bloc s'ha d'executar, fent les diferents comprovacions dins, i a partir d'una *Switch Unit* tractar aquest valor per escollir el bloc. Pot ser molt útil en el cas de tenir moltes possibilitats i no voler tenir un diagrama molt gran. A més a més, com les variables utilitzades acostumen a ser les mateixes, no tenim molts paràmetres d'entrada dins la unitat i també estalviem links.

1. Creem una *Script Unit* amb el codi necessari per a fer totes les comprovacions i que tindrà com a sortida un únic paràmetre corresponent a un codi per a cada ALT possible.
2. Creem una *Switch Unit* amb un *Case Value* per a cada codi que pugui resultar de l'*Script Unit*.
3. Unim el resultat de l'*Script Unit* amb la *Switch Unit* amb un OK Link enllaçant el paràmetre de sortida amb el *Switch* de la unitat.
4. Unim la *Switch Unit* amb la primera operació de cada ALT amb un OK Link segons el *Code* que haguem escollit per l'ALT corresponent.
5. Unim la *Switch Unit* amb la primera operació de fora dels blocs amb un OK Link sense *Code*.
6. Unim l'última operació de cada ALT amb la primera operació de fora dels blocs amb el link que faci falta.



Il·lustració 93: ALT. Codificació amb Script Unit i Switch Unit (WebML)

En aquest cas tenim tres possibles resultats des de la *Switch Unit*. El conjunt de resultats ja inclou un que és el cas de no coincidir amb cap. Per això, tractaríem el link *default* com si fos un error.

Bucles

En el cas d'aquesta aplicació no s'ha utilitzat mai el fragment d'interacció LOOP a l'especificació d'UML.

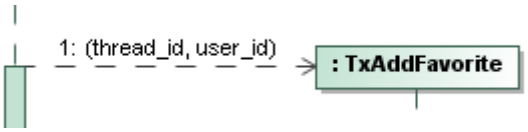
Donat aquest cas, no s'ha pogut extreure cap patró al respecte però volia remarcar l'existència de la *Loop Unit* que serveix per a iterar sobre elements d'un vector d'entrada. Possiblement aquesta unitat pugui ser utilitzada en alguns dels casos en que a UML usem el LOOP.

8.3.2.5 Creació, modificació o eliminació d'instàncies i relacions

A l'UML donàvem per suposada una implementació de persistència directa per a la capa de dades. WebML implementa un sistema similar però hem de tenir en compte que no és 100% igual.

Creació

Cada classe a l'UML té una operació creadora. Aquesta operació, genera la instància donats uns paràmetres d'entrada.



Il·lustració 94: Creació d'una instància UML)

En el patró estructural "De funcions a mòduls" veurem que podem traduir totes les funcions en un mòdul i encapsular la funcionalitat a la vegada que fem la traducció interna segons els patrons corresponents. Tot i així, durant el procés de traducció concret d'aquesta aplicació hem fusionat la funció creador amb les transaccions que la usaven. D'aquesta manera, el tractament de les variables que fèiem a la funció creadora l'hem fet a la transacció.

De totes formes, l'únic que canvia és l'encapsulament en un mòdul del que explicarem a continuació.

Per tal d'afegir una nova instància al sistema i mantenir-la a la base de dades, WebML ofereix la unitat *Create Unit*. Donada una entitat, és possible enllaçar tota la informació relativa als seus atributs i a les claus de les relacions amb cardinalitat 1. Un cop activada, es crearà la instància amb la informació que hi hagi en aquell moment.

Cal destacar que tot i tenir una relació de cardinalitat 1, si l'atribut queda buit no donarà cap error a l'hora de crear aquesta nova instància. La restricció s'ha de controlar des de domini.

A més a més, al no poder definir valors *unique* al model de dades, s'ha de tenir en compte de que si no volem un valor repetit entre les instàncies d'una entitat, ho hem de comprovar manualment abans de crear-la.

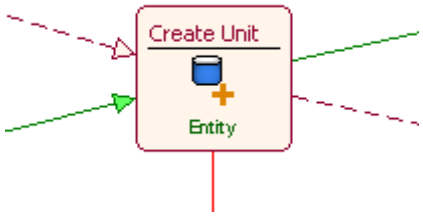
Igual que la resta d'unitats operacionals, la *Create Unit* pot tenir tants links d'entrada com faci falta per tal d'unir tots els paràmetres que vulguem.

Així doncs, per a crear la instància afegiríem una *Create Unit* de la entitat corresponent i assignaríem els valors ja tractats. És per això que s'ha comentat la possibilitat d'usar un mòdul, per als casos en els que la informació d'entrada a la creació no és directament la informació que es guarda o, com hem vist en algun cas, entra en joc el temps actual, que no té a veure amb paràmetres d'entrada i es calcula dins la funció.

1. Creem una *Create Unit* de l'entitat que vulguem crear.
2. Unim tots els links necessaris d'entrada a la unitat per tal d'enllaçar els paràmetres ja tractats per a la creació de la unitat. S'ha de tenir en compte d'enllaçar tots els

paràmetres necessaris per a no crear inconsistències a la base de dades (cardinalitats 1) i que si volem valors que no es repeteixin ho hem d'haver comprovat en una altra unitat.

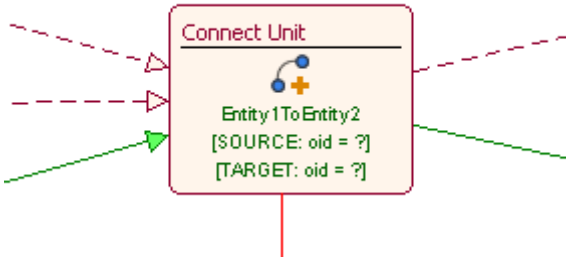
3. Unim la unitat anterior amb la *Create Unit* amb el link corresponent.
4. Unim la *Create Unit* amb la següent unitat a executar amb un OK Link.
5. Tractem amb un KO Link un possible error intern durant la creació de la instància.



Il·lustració 95: Creació d'una instància (WebML)

Per acabar, hi ha un punt que no s'ha tractat encara sobre la creació i és el de les relacions de cardinalitat N. En aquest cas, la unió no es pot fer amb una *Create Unit*. Per tal de fer-ho, s'utilitza la *Connect Unit* que permet, donada una relació, unir la instància origen i amb la instància destí de la relació. Si es vol assignar a la mateixa funció creadora, s'ha de vigilar en fer-ho després de la creació de la entitat ja que necessitarem el seu identificador.

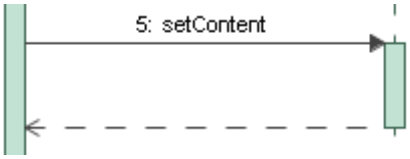
1. Creem una *Connect Unit* de la relació que volem afegir.
2. Unim amb els links necessaris els identificadors de les dues entitats per tal de poder fer la unió.
3. Unim la unitat anterior amb la *Connect Unit* amb el link corresponent.
4. Unim la *Connect Unit* amb la següent unitat a executar amb un OK Link.
5. Unim amb Transport Links la *Connect Unit* amb les unitats que necessitin els identificadors en cas de voler agafar la informació des d'aquesta unitat.
6. Tractem amb un KO Link un possible error intern durant la creació de la relació.



Il·lustració 96: Relació de dues instàncies amb cardinalitat N (WebML)

Modificació

La modificació d'instàncies es fa mitjançant les funcions *set* de cada atribut. A diferència de l'UML, on hem de fer els *set* per separat per a cada atribut, a WebML podem modificar tots els atributs que vulguem d'una sola vegada.



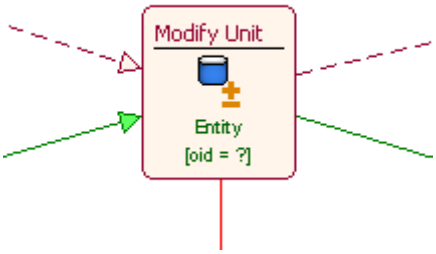
Il·lustració 97: Modificació d'un atribut d'una instància (UML)

Aquesta operació es fa mitjançant la *Modify Unit*. Donada una entitat, és possible enllaçar tota la informació relativa als seus atribut i les claus de les relacions amb cardinalitat 1. Un cop activada, modificarà els valors actuals de la instància per als enllaçats per tots els enllaços que tingui. L'elecció de la instància a modificar es fa a partir del seu identificador i el selector es genera automàticament a WebRatio.

Cal destacar que si enllacem un paràmetre i l'entrada és *null*, aquest valor serà modificat igualment sense donar cap error. Aquest fet pot crear inconsistències sobretot en les relacions de la instància i s'ha de controlar des de domini.

En el cas de voler traduir exactament el que hi ha a l'UML, haurem de fer una *Modify Unit* per a cada *set* i enllaçar només en aquesta unitat l'atribut que modifiquem a l'UML.

1. Creem una *Modify Unit* de l'entitat que volem modificar.
2. Unim el link necessari per tal d'enllaçar el paràmetre a modificar des de la unitat destí fins a la *Modify Unit*.
3. Unim la unitat que generi l'identificador de la instància a modificar amb la *Modify Unit* i l'enllacem amb la *Key Condition*.
4. Unim la unitat anterior amb la *Modify Unit* amb el link corresponent.
5. Unim la *Modify Unit* amb la següent unitat a executar amb un OK Link.
6. Tractem amb un KO Link un possible error intern durant la modificació de la instància.



Il·lustració 98: Modificació d'un atribut d'una instància (WebML)

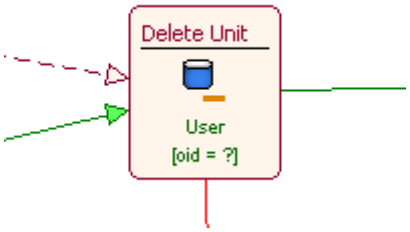
Si volem aprofitar la potència de la unitat, podem fer més d'un set a l'hora sempre i quan no modifiqui el comportament definit a l'UML enllaçant més variables d'entrada.

Eliminació

Quan eliminem instàncies a l'UML sempre hem de vigilar de deixar el sistema consistent respecte a les relacions. A WebML funciona exactament igual ja que com hem vist a la creació o a la modificació, no es generen errors en el cas de crear instàncies inconsistents.

La *Delete Unit* elimina una instància d'una entitat a partir del seu identificador. Aquest selector és generat automàticament a WebRatio.

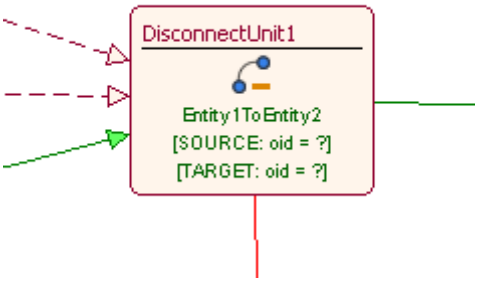
1. Creem una *Delete Unit* de l'entitat que volem eliminar.
2. Unim la unitat que genera l'identificador de la instància a eliminar amb la *Delete Unit* fent l'enllaç de l'identificador amb la *Key Condition*.
3. Unim la unitat anterior amb la *Delete Unit* amb el link corresponent.
4. Unim la *Delete Unit* amb la següent unitat a executar amb un OK Link.
5. Tractem amb un KO Link un possible error intern durant l'eliminació de la instància.



Il·lustració 99: Eliminació d'una instància (WebML)

Per tal d'eliminar les relacions de cardinalitat N haurem d'usar les *Disconnect Unit*. Aquestes unitats funcionen igual que les *Connect Unit* però produeixen l'efecte contrari.

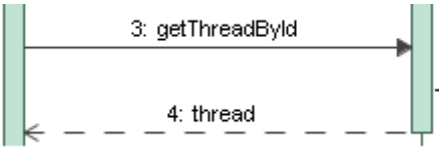
1. Creem una *Disconnect Unit* de la relació que volem desfer.
2. Unim amb els links necessaris els identificadors de les dues entitats per tal de poder desfer la unió.
3. Unim la unitat anterior amb la *Disconnect Unit* amb el link corresponent.
4. Unim la *Disconnect Unit* amb la següent unitat a executar amb un OK Link.
5. Tractem amb un KO Link un possible error intern durant l'eliminació de la relació.



Il·lustració 100: Eliminació d'una relació de cardinalitat N (WebML)

8.3.2.6 Accés a la base de dades (extracció d'instàncies)

Hi ha vegades que volem extreure instàncies de la base de dades per a tractar-les o fer servir certa informació. Per tal de poder-ho fer utilitzem la *Selector Unit*.



Il·lustració 101: Accés a la base de dades per extreure un tema de discussió (UML)

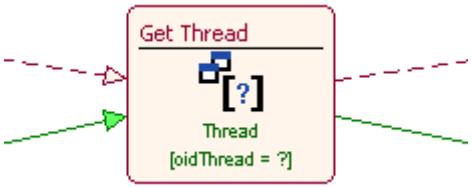
Cal notar que no utilitzarem la unitat en el cas de tenir la informació a extreure a una pàgina, sinó quan la informació és per tractar-la. Excepcions són quan utilitzem la informació de la instància per a omplir un camp d'un formulari com a valor predefinit o en un text.

Per tal de seleccionar les instàncies, utilitzem *Key Condition* si volem extreure-les a partir d'una sèrie d'identificadors i *Attribute Condition* per a la resta d'atributs que vulguem filtrar ja siguin de la pròpia instància o d'associacions.

Junt amb aquests tipus d'accessos a la base de dades, a UML acostumem a esperar una excepció en el cas de que la cerca no retorni cap resultat. Aquest no és el cas de l'ús de la *Selector Unit* ja que en cas de no trobar cap coincidència retorna un conjunt buit. Així doncs, l'excepció l'haurem de controlar amb una *Is Not Null Unit*. Aquesta part està més explicada al patró operacional "Excepcions".

1. Creem una *Selector Unit* de l'entitat de la que volem extreure les instàncies.
2. Afegim les *Key Condition* o *Attribute Condition* en el cas de que sigui necessari per a fer la selecció d'instàncies. Si no s'especifica res s'agafen totes.
3. Enllacem els orígens dels valors per a les condicions amb la unitat amb Transport Links per tal de que s'esculli la informació correcte.

Des de la unitat podem enllaçar amb d'altres per a usar el resultat consistent en conjunt d'instàncies coincidents. Podem escollir fer conjunt de qualsevol atribut de l'entitat o qualsevol atribut d'entitats relacionades.

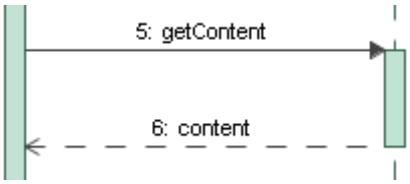


Il·lustració 102: Accés a la base de dades per a extreure un tema de discussió (WebML)

A la imatge podem veure una *Selector Unit* que extreu una instància de l'entitat Thread a partir d'una *Key Condition* sobre el seu identificador.

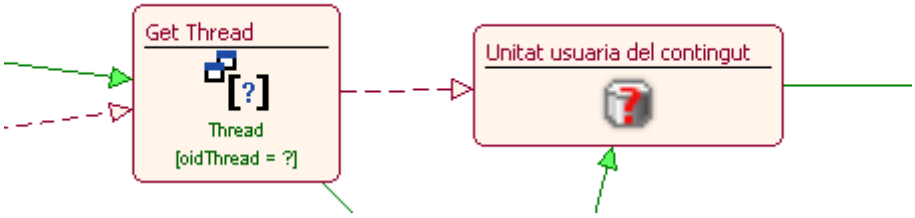
8.3.2.7 Extracció de dades d'una instància

Normalment, quan hem extret una instància de la base de dades és per agafar-ne el valor d'algun dels seus atributs. A l'UML utilitzem les funcions *get* sobre la instància amb aquest objectiu.



Il·lustració 103: Extracció de dades d'una instpancia (UML)

A WebML, però, el que fem és agafar els valors des de la unitat d'accés a la base de dades.



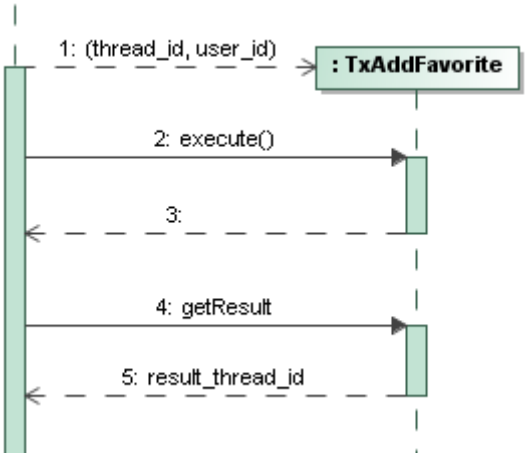
Il·lustració 104: Extracció de dades d'una instpancia (WebML)

Això és deu a que no tenim necessitat de crear cap objecte amb la informació perquè podem utilitzar els Transport Links des de l'unitat fins el lloc on els necessitem si no hem canviat el valor abans.

S'ha de tenir en compte que això serveix quan la *Selector Unit* utilitzada per a accedir a la base de dades retorna una sola instància. En cas contrari, estarem enllaçant tota la llista dels valors de l'atribut en qüestió de totes les instàncies extretes.

8.3.2.8 Ús de transaccions

Per a usar transaccions a UML, el primer que es feia era preparar-la amb la creació d'una instancià i afegint els paràmetres necessàries per al seu posterior ús. Més tard s'executava cridant l'operació *execute()* que heretaven totes al ser subclasses de Transacció i després podíem obtenir els resultats fent un *get* de l'atribut corresponent de la transacció.



Il·lustració 105: Ús d'una transacció (UML)

A WebML podem simular una cosa semblant tot i que s'ha d'entendre d'una altra manera.

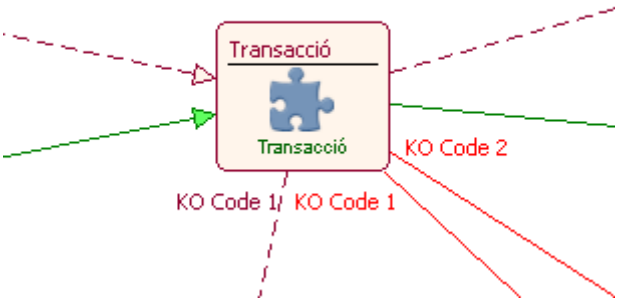
Per a usar una transacció creem una *Module unit* ja que tenim les transaccions definides com a mòduls i indiquem quin tipus de mòdul representa.

L'execució de l'operació és a través dels links com la resta d'operacions de WebML i per tant, en el seu moment d'execució ha de tenir ja els paràmetres emparellats.

Un cop executada podem extreure la informació de retorn de les seves sortides (*OK* i *KO Collectors*). Si ho fem des d'un Transport Link, haurem d'especificar de quin *Collector* extraiem la informació.

S'haurà de tenir en compte tant l'OK Link del mòdul que indica l'execució correcte com els KO Links corresponents als *KO Collector* que indicaran molt probablement una excepció de la transacció que pot ser que haguem de capturar o rellançar.

Així doncs, prendrem com a exemple una transacció amb uns paràmetres d'entrada i dos possibles excepcions i la figura mostrada anteriorment per a mostrar la següent traducció.



Il·lustració 106: Ús d'una transacció (WebML)

Com a conclusió podríem seguir els següents passos:

1. Creem una *Module Unit* del tipus de la transacció que volem executar.
2. Emparellem amb Transport Links els paràmetres necessaris.
3. Unim amb un Link l'operació anterior amb la *Module Unit* i acabem d'emparellar els paràmetres necessaris si en faltava algun a través del link.
4. Unim amb un OK Link la *Module Unit* amb la següent operació en cas d'èxit.
5. Unim amb un KO Link la *Module Unit* amb la següent operació en cas d'excepció per a tots els *Collector Code* d'error possibles.
6. Emparellem amb Transport Links els paràmetres de sortida segons el *Collector Code* necessari.

8.3.2.9 Inicialitzar valors o tractar text

Abans d'explicar aquests patrons cal dir que no s'han extret de cap cas utilitzat durant la pràctica. Són resultat del coneixement al que s'ha arribat sobre certes unitats en el transcurs de la traducció.

És possible que a vegades haguem d'inicialitzar un valor d'una variable, sobreescriure un valor de la variable per el d'una altra o un constant o per exemple concatenar dues variables de tipus *String* per a generar una cadena nova.

En aquest punt s'ha de tenir molt present una cosa sobre el funcionament de les unitats. Cada unitat té uns possibles paràmetres d'entrada i té uns possibles paràmetres que podem enllaçar. En el cas d'utilitzar una variable que modifiqui el seu valor durant l'execució, s'ha de tenir en compte d'agafar el valor de la última unitat visible que l'hagi modificat.

Inicialitzar una variable

La única manera de generar una variable del no res és a partir d'una *Script Unit*. Aquesta unitat, com ja es pot haver vist en algun altre patró, permet escriure un *script* amb el llenguatge Groovy.

Les *Script Unit* permeten paràmetres d'entrada i poden generar tants paràmetres de sortida com sigui necessari. D'aquesta manera, podem dur a terme qualsevol operació que vulguem.

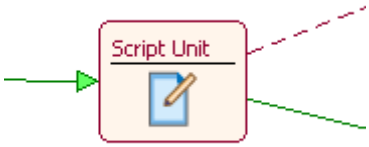
Si volem inicialitzar una variable seria tan fàcil com generar una *Script Unit* sense entrades i que com a sortida tingués la variable que volem amb el seu valor inicial.

```
#output String text
return [ "text" : "Initialized text." ]
```

En aquest exemple podem veure com generem una variable de sortida de tipus *String* que anomenem "text" i que pren el valor de "Initialized text.". Podem fer el mateix amb altres tipus de variables de la mateixa manera.
En el cas de voler inicialitzar més d'una variable alhora, només cal crear més d'un output i retornar-los de la següent forma:

```
return [ "variable1" : "Valor variable 1", "variable2" : "Valor variable 2", etc... ]
```

Aquest cas s'ha usat durant el projecte sobre tot per tal de generar els diferents missatges d'error a mostrar en cas de capturar excepcions.



Il·lustració 107: Inicialitzar valors amb Script Unit (WebML)

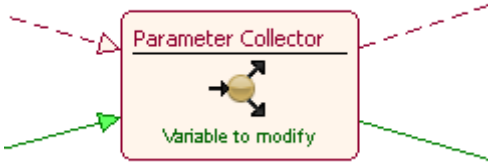
Modificar el valor d'una variable

A WebML cada unitat té unes variables possibles de sortida. És per això que a l'hora d'utilitzar una variable sempre hem d'agafar la més propera tal i com faria el compilador. Quan volem assignar un nou valor a una variable el que volem és crear una unitat on, des d'aquella, puguem agafar el paràmetre amb el mateix nom que teníem però amb un valor diferent.

Si tenim una variable A i una variable B i en cert moment volem que A tingui el valor de B, hauríem de crear una unitat des de la qual poguéssim agafar A i fos realment B. És a dir, tenir B d'entrada i A de sortida de manera que agaféssim sempre aquesta A en el moment de necessitar-la i no la A de la unitat anterior.

Aquest efecte el podem crear amb una *Parameter Collector Unit*. Aquesta unitat permet definir uns paràmetres amb els noms que vulguem i enllaçar per l'entrada qualsevol altre variable. D'aquesta manera, a partir d'aquest punt podem utilitzar qualsevol d'aquestes variables des de la unitat.

1. Creem una *Parameter Collector Unit*.
2. Afegim un *Collector Parameter* a la *Parameter Collector Unit* corresponent al paràmetre de sortida corresponent a la part esquerra de l'assignació.
3. Unim el valor de la part dreta de l'associació amb la *Parameter Collector Unit* amb el link que correspongui.
4. Unim la *Parameter Collector Unit* amb les unitats que necessitin la variable modificada amb el link corresponent.



Il·lustració 108: Modificar variables amb Parameter Collector (WebML)

Podríem utilitzar el mateix mètode per a fer el mateix a més d'una variable en comptes d'utilitzar una *Parameter Collector Unit* per a cada un afegint un *Collector Parameter* per a cada variable.

Concatenació d'Strings i altres operacions

Com ja hem vist en el cas d'inicialitzar una variable, les *Script Unit* tenen molt de potencial. Tenint en compte que podem fer qualsevol operació que Groovy permeti, podem aprofitar aquesta unitat per a concatenar cadenes, fer sumes de valors d'un vector o qualsevol cosa relativament senzilla amb poques línies.

Per tal de fer-ho hauríem d'entrar en més detalls del llenguatge Groovy i no és part d'aquest projecte però comentarem el sistema d'*inputs* i *outputs* per tal de saber com utilitzar la *Script Unit* independentment del procés intern.

S'ha de definir cada paràmetre d'entrada a la unitat de la forma `#input TipusVariable NomVariable`. D'aquesta manera, a l'unir una altra unitat amb la *Script Unit* podem enllaçar aquests paràmetres si el link és d'entrada.

I com ja hem vist a l'inicialitzar variables, els valors de sortida es defineixen de la forma `#output TipusVariable NomVariable`. Aquests podran ser enllaçats quan s'uneixi la *Script Unit* amb altres unitats amb links de sortida.

Cal destacar que no s'ha utilitzat durant el projecte la *Script Unit* amb aquesta finalitat tot i que s'ha trobat aquest ús a l'entendre com funciona la unitat.

8.3.2.10 Temps

El tractament del temps a UML es fa suposant una sèrie d'operacions globals com ara *now()* o *today()* que retornen en cada cas una variable de tipus *timestamp* o *date* que utilitzarem dins dels nostres models.

A WebML tenim les *Time Unit* que permeten simular un comportament semblant i extreure certa informació sobre la data i temps actuals.

Les opcions que ofereix de sortida són:

- Current/Input Date
- Current/Input Time
- Current/Input Timestamp
- Day
- Hour
- Minutes
- Month
- Next Midnight Timestamp
- Seconds
- Time in Milliseconds
- Year

És possible enllaçar una variable de tipus *date, time* o *timestamp* a l'entrada de la unitat per extreure informació d'altres moments.

1. Creem una *Time Unit*.
2. Unim, si cal, una unitat amb la *Time Unit* amb un Transport Link per modificar el temps actual per a un altre.
3. Unim la unitat anterior amb la *Time Unit* amb el link que correspongui.
4. Unim la *Time Unit* amb la següent operació amb un OK Link.
5. Enllacem el paràmetre de sortida que vulguem amb el paràmetre d'una altra unitat a través d'un Transport Link o de l'OK ja creat.

Il·lustració 109: Time Unit (WebML)

8.3.2.11 Sessió

L'ús de la sessió a WebML està controlat per tres unitats:

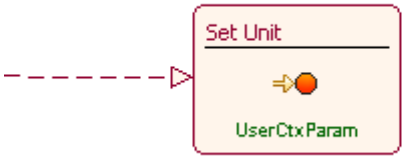
- Set Unit
- Get Unit
- Reset Unit

Aquestes tres unitats no accepten entrades de OK ni KO Links. Per usar-les cal que arribi o surti un Transport Link i en el moment que la unitat relacionada s'activi, l'operació relativa a la sessió també s'executarà.

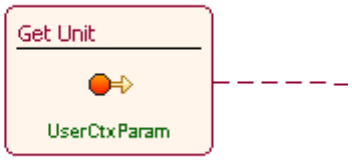
Per tal de traduir l'ús de la sessió d'UML a WebML és tan fàcil com substituir els *set* d'una variable per una *Set Unit*, *get* per una *Get Unit* i el fet d'esborrar una variable amb la *Reset Unit*.

A través del Transport Link podem enllaçar els paràmetres que vulguem guardar o extreure de la sessió.

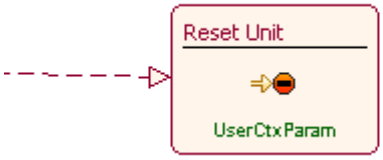
A continuació es mostra l'aparença de cada una de les unitats comentades.



Il·lustració 110: Set Unit (WebML)



Il·lustració 111: Get Unit (WebML)



Il·lustració 112: Reset Unit (WebML)

En tots els casos, s'utilitzarà com a *Context Parameter* de les unitats el valor UserCtxParam.

8.3.3 Estructurals

8.3.3.1 Del diagrama intern a pàgines i mòduls

Crear l'estructura de l'aplicació en una *site view* és molt senzill tenint en compte la següent idea: les *server page* són pàgines i els *business helper* són mòduls. Això ho hem pogut veure a l'hora de traduir totes les pàgines i *business helper* en punts anteriors i ha funcionat de forma correcta.

A partir d'aquí per a cada *server page* generarem una pàgina a la *site view* amb el nom de la *server page* o a la pàgina del client a la que representa de l'UX Model.

De la mateixa manera, per a cada *business page* generarem un nou mòdul dins dels mòduls de l'aplicació. Per comoditat mirem de separar-los dels de les transaccions. Un cop fets crearem una *Module Unit* dins la *site view* representant al *business helper*.

La manera d'omplir aquestes unitats s'expliquen en altres patrons estructurals. Per el cas de la pàgina tenim el patró operacional "D'screens d'UX Model a pàgines WebML" i per als *business helper* el patró estructural "De *business helper* a mòduls".

8.3.3.2 De business helper a mòduls

Tal com s'ha pogut veure a l'hora de traduir el diagrama intern de l'aplicació, s'ha decidit generar els *business helper* com a mòduls de l'aplicació. D'aquesta forma encapsulem el seu comportament, queda més net i pot ser reutilitzat en diferents vistes de l'aplicació creant una nova instància. Destacar que aquests mòduls, per comoditat, es podien posar a una vista diferent dels de les transaccions.

En l'apartat de traducció es va explicar que havia separat les responsabilitats del *business helper* per a la naturalesa de les unitats de WebML. D'aquesta forma les feines corresponents a les *content unit* es veien desplaçades a les pàgines de WebML, que podrien ser relacionades amb les *server page* del diagrama.

Les responsabilitats que deixarem al *business helper* les podem classificar en els següents grups:

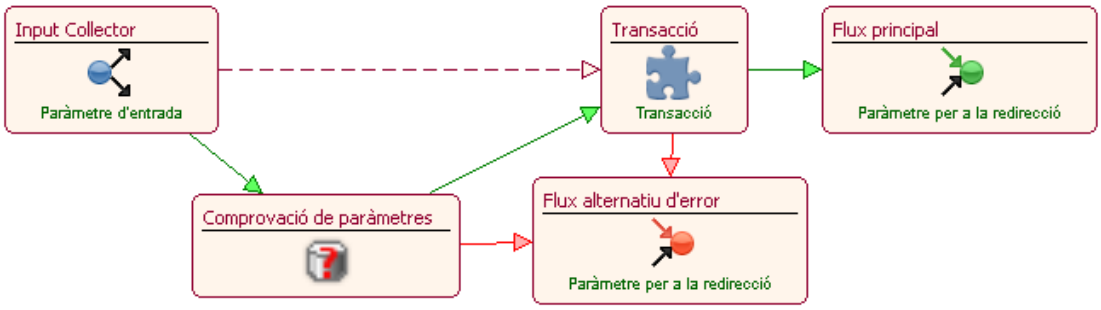
- Comprovació de paràmetres
- Extracció d'informació per a omplir camps d'un formulari
- Utilització de les transaccions
- Redirecció segons els resultats

Donada la mateixa estructura a l'UML prosseguirem amb els passos a seguir per a la traducció.

1. Creem un mòdul dins de la *model view* corresponent als *business helper* amb el nom del business helper al diagrama intern de l'aplicació d'UML després d'aplicar l'extensió web.
2. Creem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres amb els que pot ser que es cridi amb algun link des de les pàgines.
3. Fem la comprovació dels paràmetres de l'*Input Collector* segons el patró operacional corresponent.
4. Creem una *Model Unit* per a cada transacció que volem utilitzar del tipus corresponent.
5. Per al cas base de continuïtat de les operacions (pàgina de destí usual) utilitzarem un *OK Collector* que tindrà el mateix nom de la sortida al diagrama intern de l'aplicació (p.e. nom del *business helper* OK). En el cas de que l'última operació no enllaci amb un OK Link amb l'*OK Collector*, creem un *No Op Operation Unit* que enllaçarem amb el link corresponent (KO Link) i unirem aquesta nova unitat amb l'*OK Collector* amb l'OK Link.
6. Per a la resta de possibles vies que acostumen a ser redireccions utilitzarem un *KO Collector* per a cada una d'elles amb el mateix nom de la sortida al diagrama intern de l'aplicació.
   Si ha de transportar informació, s'afegirà un *Output Collector Parameter* per a cada un dels paràmetres a transportar.
7. Afegim un *Output Collector Parameter* per a cada un dels paràmetres que són necessaris transportar a la *Server Page*. Molt probablement serà tota la informació extreta en les transaccions o part d'aquesta.

Cal fer notar que en el cas de que en el punt 1 ja existeixi tal mòdul dins l'aplicació però està buit, reutilitzem aquest en comptes de crear-ne un nou. Això significa que s'ha fet un mòdul

placeholder perquè en algun moment s'ha necessitat per a modelar l'aplicació, per exemple a les *site views*.



Il·lustració 113: Estructura d'un business helper (WebML)

El flux de les operacions està marcat pels OK i KO Links però els paràmetres de l'*Input Collector*, excepte en la primera operació, estaran transportats amb Transport Link des de l'*Input Collector* fins a la unitat que els necessiti.

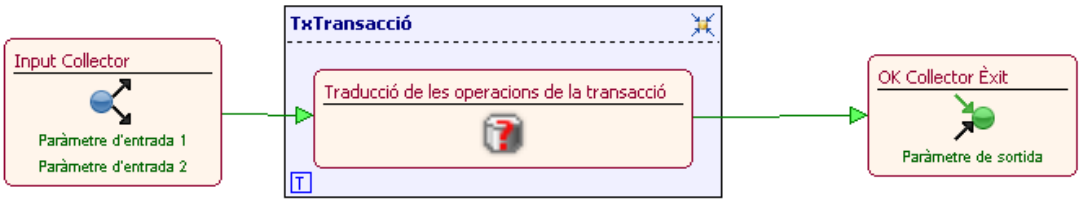
Les unions amb les pàgines o altres *business helpers* es fan des de fora del mòdul i només s'han de tenir en compte per a fer els *OK* i *KO Collector* corresponents.

8.3.3.3 De transaccions a mòduls

Durant la traducció de l'aplicació hem traduït tots les transaccions com a mòduls. D'aquesta forma hem pogut generar les operacions una vegada i poden ser reutilitzades de la mateixa manera que les transaccions a l'UML.

Com ja hem vist, s'ha usat l'*Operation Group* per a donar el valor transaccional a les operacions i avortar la funcionalitat en cas d'error sense generar inconsistències. D'aquesta manera en el moment llançar possibles excepcions només cal sortir amb un KO Link de l'*Operation Group* i això aturarà la transacció. A més a més, si l'unim a un *KO Collector* podem controlar l'excepció al usar el mòdul.

1. Creem un *Input Collector* on afegirem un *Input Collector Parameter* per a cada un dels paràmetres d'entrada de la transacció.
2. Creem un *Operation Group* on s'inclouran les operacions de la transacció.
3. Segons el patró operacional que correspongui traduïm les operacions internes de la transacció.
4. Utilitzem un *OK Collector* com a sortida de la transacció en cas d'èxit. Si la transacció té valors de sortida, els afegim com a *Output Collector Parameter* a l'*OK Collector*.
5. Enllacem l'última operació amb l'*OK Collector* amb un OK Link. En el cas de no poder utilitzar l'OK Link, creem un *No Op Operation Unit* que enllaçarem amb el link corresponent (KO Link) i unirem aquesta nova unitat amb l'*OK Collector* amb l'OK Link.



Il·lustració 114: Estructura d'una transacció (WebML)

Els *OK Collector* i els *KO Collector* que puguin aparèixer al traduir les operacions aniran fora de l'*Operation Group*.

8.3.3.4 De funcions a mòduls

Quan utilitzem una transacció a UML sempre ho fem assignant els paràmetres i després cridant la funció *execute()* que fa les operacions pertinents segons la transacció.

Igual que hem pogut modelar el comportament d'aquesta funció seguint els passos explicats al patró estructural "De transaccions a mòduls" podem fer-ho amb la resta de funcions.

9 Validació

El procés de validació dels patrons de traducció consistia en fer una petita especificació en UML d'una nova aplicació per tal d'intentar fer la traducció usant els patrons definits i veure que realment són correctes.

A mesura que es modelaven els diferents casos d'ús del sistema a WebML s'ha pogut anar veient el funcionament i la forma de fer-ho, de manera que després de varis casos d'ús iguals era molt fàcil fer-ne un de nou ja que els patrons s'havien pogut identificar tot i que no estiguessin completament escrits.

Podríem dir que amb cada cas d'ús s'ha anat millorant la concepció del patró tot i que no s'ha provat de fer-ho sense cap altre coneixement que els patrons, que seria probablement la millor forma de veure que funcionen.

És a dir, la millor manera de poder fer la validació seria que una altra persona sense experiència utilitzés els patrons per a veure quin nivell de coneixement de WebML és necessari per a poder-los utilitzar i si una persona amb coneixements nuls o molt bàsics seria capaç de fer les traduccions. D'aquesta forma la valoració seria més objectiva.

Aquesta opció és la que es proposa a treballs futurs. D'aquesta forma, mentre es valida també es podrien extreure nous patrons en els casos en que cap dels especificats es pugui aplicar.

10 Planificació final

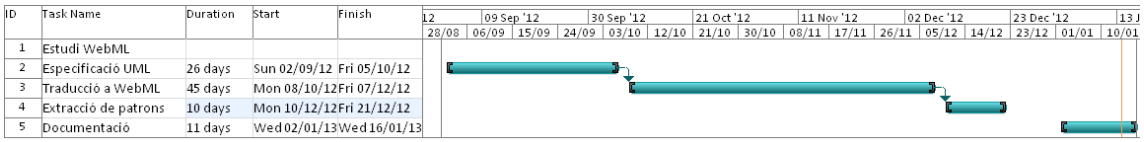
A mesura que es duia a terme el projecte em vaig adonar d'una cosa. Havia estat molt optimista a l'hora d'assignar el temps a les diferents tasques.

La traducció a WebML va ser més dura de l'esperat. A mesura que avançava amb la traducció m'anava donant compte d'errors que havia comès a altres diagrames i havia de tornar a fer modificacions a tot l'anterior.

A més a més, l'extracció de patrons ha estat molt difícil de documentar per a poder trobar una bona estructura i justificació.

I per acabar, la validació s'ha vist redundant ja que no ajudava realment a validar els patrons en el cas de que la fes un mateix.

En definitiva, el temps dedicat a les diferents tasques va quedar repartit de la següent manera:



ID	Task Name	Duration	Start	Finish
1	Estudi WebML			
2	Especificació UML	26 days	Sun 02/09/12	Fri 05/10/12
3	Traducció a WebML	45 days	Mon 08/10/12	Fri 07/12/12
4	Extracció de patrons	10 days	Mon 10/12/12	Fri 21/12/12
5	Documentació	11 days	Wed 02/01/13	Wed 16/01/13

Il·lustració 115: Gantt de la planificació final del projecte

Per tal de fer la valoració econòmica del projecte, contarem el nombre d'hores total que s'han treballat.

Per tal de valorar el projecte i estudiar el funcionament de WebRatio es van invertir un total de 150h per a tenir uns coneixements inicials per a poder començar el projecte.

Des de setembre fins finals de desembre es van treballar 400h en l'especificació i el desenvolupament de l'aplicació.

A més a més, s'han de sumar les hores de documentació que han estat unes 80h.

En total es comptabilitzen unes 630h de feina per tal de poder extreure els patrons de traducció. Si agafem un cost de 45€/h, el cost total del projecte són **28350€**.

11 Conclusions

11.1 Facilitat de traducció

Traduir d'UML a WebML no és molt complicat però com s'ha pogut veure en els patrons extrets, hi ha moltes casuístiques que fa complicat intentar definir una forma sistemàtica de traduir qualsevol model.

De totes formes, WebML ajuda a modelar sobre tot el contingut que tindrà una pàgina i és molt fàcil d'usar en aquest aspecte.

11.2 Extracció de patrons

En alguns casos ha estat difícil extreure els patrons de traducció perquè tot i ser molt semblants, hi havia certs aspectes que diferenciaven un cas de l'altre i s'ha hagut d'anar en compte de no ajuntar coses que no es podien traduir de la mateixa forma.

A més a més, com ja era d'esperar, no s'ha aconseguit fer un conjunt de patrons complets per a traduir qualsevol model ja que l'exemple no era complet. Com es proposa per a treballs futurs, si es vulgues seguir amb l'extracció de patrons s'haurien de buscar exemples d'un altre tipus, potser amb més lògica interna, per a poder trobar casos diferents que no s'han vist en el cas del fòrum. Tot i així, mai s'ha de deixar de banda l'objectiu de WebML que són les aplicacions centrades en gran volum de dades.

11.3 Quan usar WebML/WebRatio i no UML

Tot i que no és complicat traduir d'UML a WebML, des del meu punt de vista, és quelcom redundant. Fent això tenim la mateixa especificació feta amb dos models diferents que intenten comportar-se de la mateix forma.

A partir d'aquí, recomanaria usar WebRatio directament en el cas de voler fer una aplicació web de forma incremental, com proposen algunes metodologies àgils, ja que permet tenir sempre l'especificació al mateix nivell que l'aplicació i en qualsevol moment generar el codi de l'especificació escrita fins el moment. De fet, aquest és un dels objectius de WebRatio, modelar processos i poder implementar el sistema per a complir certs requisits i objectius.

Tot i això, la corba d'aprenentatge pot ser un problema inicial ja que hi haurà casos on costarà més modelar amb WebRatio que amb UML però pot valer la pena ja que els canvis posteriors només s'hauran de fer al model ja que el codi estarà generat automàticament.

Els principals problemes que es podrien trobar a l'hora de fer servir WebML seria el modelatge de webs amb molta lògica interna i processos més elaborats tot i que, pel que he pogut veure fins el moment, les *Script Unit* semblen una bona opció per a aquests casos.

A més a més, existeix la possibilitat d'usar WebML com un llenguatge de modelatge i oblidar-se de la possible generació de codi. D'aquesta manera podríem modelar qualsevol aplicació web de la mateixa forma que fèiem amb UML i després generar el codi a partir d'aquesta especificació amb qualsevol llenguatge.

12 Treballs futurs

12.1 Validació per part d'una altra persona

Tal i com s'ha comentat a l'apartat de validació dels patrons, no tenia gaire sentit fer la validació tal i com s'havia presentat en un principi ja que fer la validació jo mateix no hauria aportat més del que ja aportava fer varis casos d'ús.

Per veure realment la seva utilitat hauria de fer la traducció una altra persona ja que d'aquesta manera es podria valorar el coneixement mínim de WebML per tal de poder utilitzar-los i veure si estan realment definits.

Estaria bé poder generar una altra especificació on s'intentés fer la traducció utilitzant els patrons que s'han extret en el projecte i veure si realment poden aplicar-se correctament en altres contexts i a més a més poder-ne extreure de nous en els casos de que els actuals no siguin suficients.

12.2 Ampliar els patrons de traducció

Com hem pogut veure a les conclusions, els patrons de traducció que s'han trobat durant el projecte eren dependents de l'especificació del fòrum d'aprenentatge cooperatiu.

Per tal de poder trobar-ne de nous o millorar els ja trobats, seria bo poder repetir el projecte amb altres especificacions de forma que sortissin altre tipus de patrons donat per l'ús de noves unitats que no han fet falta en aquest cas com podrien ser els *loops*, webservices, l'enviament d'e-mails, els *stored procedures*, les operacions programades per a un temps concret o la creació d'unitats pròpies amb les *custom units* per a utilitats reusables.

12.3 Capa de presentació

Un cop ja es puguin traduir els projectes seria qüestió d'estudiar el sistema de *layouts* que incorpora WebRatio per tal de poder utilitzar-lo per a generar els webs de forma automàtica.

Els *layouts* permetrien fer coses com ara links des d'informació mostrada d'una unitat, amagar cert contingut en algunes condicions (links o unitats senceres) o millorar l'aspecte de l'aplicació resultant.

Això ens permetria aprofitar tota la potència de WebRatio i poder crear prototips de forma ràpida.

13 Glossari

- **Business Helper**

Classe del diagrama intern de l'aplicació un cop aplicada l'extensió web encarregada d'interactuar amb el domini del sistema per a extreure la informació necessària per a una *Server Page*.

- **Connect Unit**

Unitat operacional que permet relacionar dues instàncies de dues entitats relacionades a partir dels seus identificadors.

- **Coupling**

Característica dels links contextuals que permet enllaçar paràmetres de sortida d'una unitat amb paràmetres d'entrada d'una altra a partir del link.

- **Create Unit**

Unitat operacional que permet generar instàncies d'una entitat concreta amb els paràmetres que tingui enllaçats d'entrada.

- **Data Unit**

Unitat de contingut que permet mostrar informació d'una instància d'una entitat i les seves relacions a la pàgina que la contingui.

- **Disconnect Unit**

Unitat operacional que permet desfer la relació entre dues instàncies de dues entitats relacionades a partir dels seus identificadors.

- **Entry Unit**

Unitat de contingut que permet agregar-hi camps per a crear un formulari a la pàgina que la contingui.

- **Groovy**

Llenguatge de programació amb el que s'especifiquen les *Script Unit* a WebRatio.

- **Index unit**

Unitat de contingut que permet mostrar informació de diferents instàncies d'una mateixa entitat i les seves relacions en forma de llista a la pàgina que la contingui.

- **KO Collector Unit**

Unitat operacional que indica un punt de sortida d'un mòdul i pot seguir-se amb un KO Link en el moment d'usar-lo.

- **KO Link**

Tipus de link que acostuma a indicar un error en l'execució d'una unitat o un resultat negatiu. El comportament depèn de la unitat origen.

- **Link**

Enllaç entre dues unitats. Pot transportar informació (Link contextual) o no (Link no contextual).

- **Math unit**

Unitat operacional que permet avaluar expressions aritmeticològiques a partir d'operands.

- **Modify unit**

Unitat operacional que permet modificar els valors dels atributs d'una instància d'una entitat a partir del seu identificador.

- **Mòdul**

Encapsulament d'unitats representant una funcionalitat.

- **Module Unit**

Unitat operacional que permet utilitzar qualsevol mòdul del sistema.

- **Normal Link**

Tipus de link entre unitats que té com a origen un element d'una pàgina representa un link clicable per l'usuari.

- **OK Collector Unit**

Unitat operacional que indica un punt de sortida d'un mòdul i pot seguir-se amb un OK Link en el moment d'usar-lo.

- **OK Link**

Tipus de link que acostuma a indicar l'èxit en l'execució d'una unitat. El comportament depèn de la unitat origen.

- **Output Parameter**

Paràmetre que s'afegeix a un KO o OK Collector per a passar variables fora de la unitat durant el seu ús.

- **Parameter Collector**

Unitat que permet rebre paràmetres d'entrada i crear-ne de sortida.

- **Script Unit**

Unitat operacional que representa un codi en Groovy que s'executa en el moment de l'execució de la unitat.

- **Selector**

Atribut de les unitats que permeten filtrar instàncies d'una entitat.

- **Selector Unit**

Unitat operacional que extreu les instàncies d'una entitat de la base de dades que compleixen amb el selector assignat.

- **Server Page**

Classe del diagrama intern de l'aplicació un cop aplicada l'extensió web encarregada de generar

- **Site View**

Conjunt de pàgines i operacions que modelen la part visible i el comportament de l'aplicació per a un grup d'usuaris.

- **Switch Unit**

Unitat operacional que permet la redirecció del flux d'execució segons el valor d'una variable d'entrada.

- **Passing**

Característica dels links contextuals que permet passar paràmetres de sortida d'una unitat amb a una altra a partir del link de forma que en la unitat destí també poden utilitzar-se com a paràmetres de sortida.

- **Power Index Unit**

Millora de l'I*ndex Unit* que permet restringir el nombre d'instàncies mostrades a la vegada que genera els links necessaris per la paginació

- **Transport link**

Tipus de link que permet transportar informació d'una unitat a una altra sense necessitat de modificar el flux d'operacions.

- **Unitat**

Component mínim de la representació d'un model a WebML.

14 Bibliografia

Alur, Deepack S. et al. (2003). *Core 2JEE Patterns: Best Practices and Design Strategies*. 2nd Edition. New York: Prentice Hall.

Brambilla, Marco. WebML: the Web Modeling Language - tutorial with audio and slides.

Disponible a: http://dbgroup.como.polimi.it/brambilla/webml

Consultat: 03-gener-2013.

Conallen, Jim (2002). *Building Web Aplications with UML*. 2nd Edition. Boston: Addison-Wesley.

Fòrum oficial de WebRatio.

Disponible a: http://forum.webratio.com/

Consultat: 10-gener-2013.

Groovy.

Disponible a: http://groovy.codehaus.org/

Contultat: 12-gener-2013.

WebRatio WebML User Guide.

Disponible a: http://downloads.webratio.com/6.1/WebRatio_WebML_User_Guide.pdf

Consultat: 13-gener-2013.

WebRatio – WebML Wiki.

Disponible a: http://wiki.webratio.com

Consultat: 06-desembre-2013.